

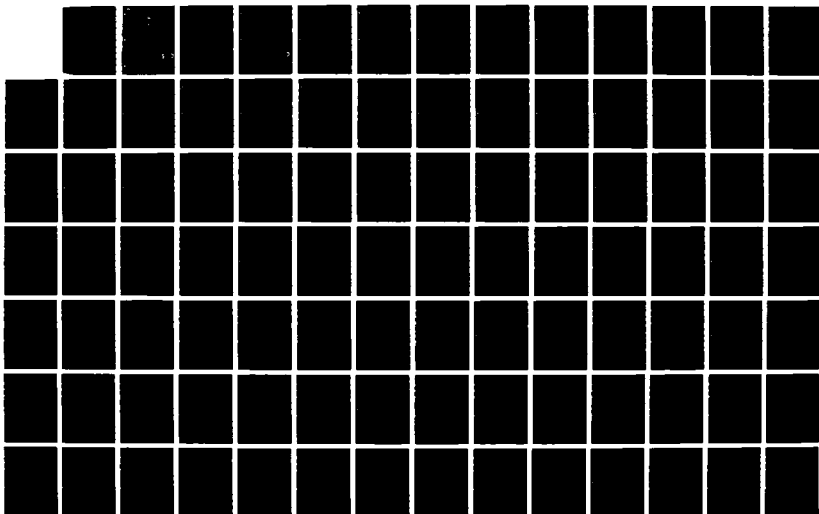
AD-A189 681

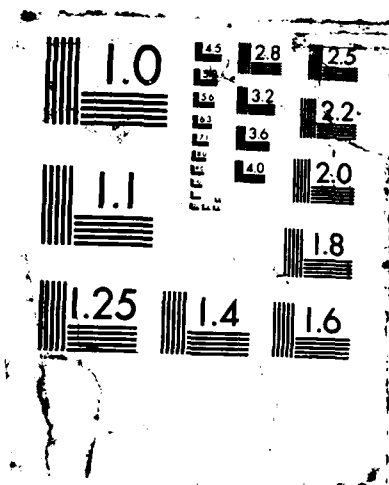
DEVELOPING A DATABASE MANAGEMENT SYSTEM AND AIR
SIMULATION SOFTWARE FOR T. (U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. M D BROOKS
DEC 87 AFIT/GCS/ENG/87D-6 F/G 15/6

1/1

UNCLASSIFIED

NL





DTIC FILE COPY

AD-A189 681



Developing a Database Management System and Air
Simulation Software for the Theater War Exercise

THESIS

Michael Drew Brooks
Captain, USAF

AFIT/GCS/ENG/87D-6

DTIC
ELECTE
MAR 07 1988
S H D

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

88 3 01 064

Preface

The goal of this thesis was to re-host the Theater War Exercise (TWX) from a mainframe to a microcomputer environment in order to incorporate modern software and hardware capabilities. To accomplish this conversion, a new TWX relational database management system was developed which would support a more "user friendly" TWX user interface.

The Theater War Exercise is a computer assisted, theater level, airpower employment exercise conducted by the Air Force Wargaming Center. TWX is designed to provide senior Air Force officers with a realistic, five day, European theater conflict which will test and evaluate their warfighting skills.

This thesis presents a background of the Theater War Exercise and the problems associated with the current Honeywell version. Additionally, the issues concerned with transporting a largescale exercise program from a mainframe to a microcomputer environment are addressed. Its main focus is on the process of replacing the current application dependent database system with a commercial DBMS and using a fourth generation applications development tool to develop a TWX controller application. Also, issues associated with converting the Fortran air battle simulation are discussed.

As in any largescale project, one must rely on the help of others. As such, I am deeply indebted to my thesis advisor, Capt Mark Roth, who's knowledge and understanding of the Theater War Exercise greatly impacted the success of this thesis effort. Additionally, I would like to thank the other members of my committee, Colonel James Weaver, Dean J. S. Przemieniecki, and Major Dan Reyen for their valuable input to this project. Finally, I would like to thank my wife, Jeannie, for her unwavering support and understanding throughout this project.

Michael Drew Brooks



Session For
S GRA&I
TAB
nounced
ification

Distribution/	
Availability Code	
Dist	Avail and/or Special
A-1	

AFIT/GCS/ENG/87D-6

Developing a Database Management System and Air
Simulation Software for the Theater War Exercise

THESIS

Michael Drew Brooks
Captain, USAF

AFIT/GCS/ENG/87D-6

DTIC
MAR 07 1988
S H D

Approved for public release; distribution unlimited

AFIT/GCS/ENG/87D-6

Developing a Database Management System and Air Simulation Software for the
Theater War Exercise

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the
Requirements for the Degree of
Master of Science (Information Systems)

Michael Drew Brooks, B.S.

Captain, USAF

December, 1987

Approved for public release; distribution unlimited

Table of Contents

	Page
Preface	ii
Table of Contents	iii
List of Figures	vi
List of Tables	vii
Abstract	ix
I. Introduction	1
1.1 Background	1
1.2 Problem Statement	6
1.3 Justification	6
1.3.1 Current System Constraints.	6
1.3.2 Project Objectives.	7
1.4 Assumptions	9
1.5 Scope	9
1.6 Development Approach	10
1.7 Material and Equipment	11
1.8 Sequence of Presentation	12
II. Software and Hardware Selection Issues	13
2.1 Rationale for Using a DBMS	13
2.2 Database Models	15
2.2.1 Relational Model.	16
2.2.2 Extended-Network Model.	17
2.3 DBMS Evaluation Methods	19

	Page
2.3.1 BIS Applied Systems, Ltd Approach.	19
2.3.2 CLEC System.	20
2.4 Micro-TWX DBMS Requirements	20
2.5 Micro-TWX DBMS Selection	22
2.6 Hardware Selection	23
III. Micro-TWX Database Design and Implementation	26
3.1 Design Approach	26
3.1.1 Entity-Relationship Process.	28
3.1.2 E-R to Relation Conversion.	32
3.1.3 Data Dependency and Normalization Process.	35
3.2 Micro-TWX Database Implementation	36
3.2.1 Micro-TWX System Layout.	39
IV. Micro-TWX Air Battle Simulation Software Conversion	41
4.1 Air Battle Simulation Software Overview	41
4.2 Simulation Conversion Process	44
4.2.1 Converting the Mission Targeting Scheme.	47
4.3 Air Battle Simulation Validation Process	49
V. Micro-TWX Controller Application	51
5.1 Functional Requirements	51
5.2 Design Approach	55
5.2.1 Design Overview.	55
VI. Conclusion and Recommendations	62
6.1 Summary	62
6.2 Recommendations for Further Work	64
A. Micro-TWX E-R Diagram	66

	Page
B. Relations Derived from E-R Diagram	70
Bibliography	81
Vita	82

List of Figures

Figure	Page
1. Central European Command and Control Structure	3
2. AAFCE Player Roles	4
3. ATAF Player Roles	5
4. Comparison of Micro-TWX and Honeywell TWX	8
5. Sample data as depicted by the Relational model	16
6. Sample data as depicted by the Extended-Network model	18
7. Micro-TWX DBMS Configuration	24
8. Micro-TWX Hardware Configuration	25
9. Partial Micro-TWX E-R Diagram	29
10. Sample E-R Diagram Showing Mapping Cardinalities	31
11. Sample E-R Diagram for "IS-A" Construct	33
12. Sample Tables Constructed From the TWX E-R Diagram	34
13. Partial Airbase Table Denoting Relation Keys	36
14. Example of the Horizontal Partitioning of the Airbase Table	37
15. Table of Ingres Indexing structures	38
16. Micro-TWX System Layout	40
17. Mission Types Modeled by the Air Simulation	42
18. Air Simulation Functional Areas	42
19. Air Simulation General Flowchart	43
20. Example Showing Ingres Embedded DBMS Calls	46
21. TWX Air Constants Table	47
22. Current TWX Mission Input Format	48
23. New TWX Mission Input Format	48
24. Micro-TWX Controller Application Main Menu	52
25. Backup and Restore Menu	59
26. Monitor Seminar Table with Sample Data	60

List of Tables

Table	Page
1. Airbase Relation	70
2. Valid Airbase Flydays	71
3. Aircraft Relation	71
4. Cargo Aircraft Relation	71
5. Recce Aircraft Sensor Relation	72
6. Aircraft on Airbase Relation	72
7. Aircraft Allowed on Airbase Relation	72
8. Valid Aircraft Rerole Relation	72
9. Preferred Munition Load Relation	72
10. Destructive Index Relation	73
11. POL, Munitions, and Spares Relation	73
12. PMS on Airbase Relation	73
13. Airbase PMS Predirect Rate Relation	73
14. Standard Munition Load Relation	73
15. Corps Relation	74
16. Land Unit Relation	74
17. Land Unit Commands Relation	75
18. Tied Land Units Relation	75
19. Land Unit Engagement Relation	75
20. Valid Land Unit Actions Relation	76
21. Land Unit Vehicles Relation	76
22. Land Unit and Quadrants Orders Relation	77
23. Seminar Control Relation	77
24. Quadrants Relation	78
25. Weather Relation	78

Table	Page
26. Air Mission Relation	78
27. Valid Mission Type - A/C Role Relation	79
28. OCA, IND, BAI, REC Escort Air Mission Relations	79
29. OCA Air Mission Targets Relation	79
30. CAS Supports Corps Air Mission Relation	80
31. BAI and IND Targets Air Mission Relations	80
32. Air and Land Constants Relations	80

Abstract

The Theater War Exercise (TWX) is a computer assisted, theater level, airpower employment exercise conducted by the Air Force Wargaming Center at Maxwell AFB, AL. Player decisions required for this exercise are typical of those that an air component commander and staff would make during an actual war. These decisions are fed into the TWX air and land battle simulation programs which model the employment of the airpower strategy, doctrine, and warfighting principles inherent in those decisions. Once the simulation is run, the players receive their battle results, logistic status, weather forecast, and new air/land orders of battle in a computer run format.

In its present configuration, the exercise runs on a Honeywell H6000 series computer and can support multiple seminars concurrently. All user inputs are made on slow, hard copy devices which severely limits the interaction by controllers and exercise participants. Also, due to an application dependent file management system and hardcoded simulation constraints, the exercise is difficult to maintain and enhance.

The goal of this thesis effort was to to bring the TWX system up to modern standards in both hardware and software capabilities while providing better exercise portability. This was accomplished by incorporating a relational database management system (Ingres) to provide increased data flexibility and a fourth generation language (4GL) to develop a new user interface. Also, in order to provide better portability, the exercise was re-hosted to an IBM PC compatible microcomputer / DEC MicroVAX II configuration. By using the Ingres network software, the PC microcomputer can run the user interface software while the MicroVAX serves as a fast database fileserver and simulation host. <

An integrated database design approach, incorporating both top-down and bottom-up relational design methods, was used to develop the new TWX database. With this approach, the top-down method of using conceptual design tools such as entity-relationship diagrams has been

combined with the traditional data dependency and normalization (bottom-up) approach. This integrated relational design method was used with great success because it allows the designer to focus on the logical database scheme rather than the physical implementation.

Next, the Fortran air battle simulation was rewritten to incorporate the Ingres DBMS embedded database calls to access the new TWX database. Also, a new mission targeting format was implemented which reflected a "real world" approach to mission planning for aircraft sorties. Finally, a TWX controller application was developed using the Ingres Application-By-Forms application development tool. This application provides a user friendly environment from which the game controller can initiate, monitor, and maintain multiple seminar exercises. By incorporating a commercial database system, improving the user interface, and re-hosting the exercise to a micro-computer environment, the new Micro-TWX system is a significant improvement over the current Honeywell system and will be much easier to maintain and enhance.

Developing a Database Management System and Air Simulation Software for the Theater War Exercise

I. Introduction

1.1 Background

The Theater War Exercise (TWX) [3] is a computer assisted, theater level, airpower employment exercise conducted as part of the Combined Air Warfare Course at the Air War College, Maxwell AFB, AL. TWX is designed to provide senior Air Force officers with a realistic, five day, European theater conflict which will test and evaluate their warfighting skills.

TWX was initially developed as a joint effort between HQ AU Data Automation¹ and the Air War College in response to a USAF Chief of Staff tasking to develop "...rigorous courses of study instructing operators and planners in the threat and application of force" [3]. As a result, the original Theater War Exercise was conducted in the spring of 1977 and considered a complete success. Since then, TWX has been incorporated as part of the Air War College's employment curriculum and is an integral element of the Combined Air Warfare Course. Also, adding to its reputation as a comprehensive educational tool, TWX has been adopted as part of the Royal Air Force Staff College and Canadian Forces Command and Staff College curriculums.

Presently, the TWX system is hosted on a Honeywell 6000 mainframe computer and can support multiple seminars (exercises) simultaneously. A typical seminar would be composed of a blue team, representing NATO forces, a red team representing Warsaw Pact forces, and a centralized game controller. The purpose of the game controller is to oversee and conduct each exercise by providing instructions, game materials, and Higher Headquarters (HHQ) guidance. The blue team

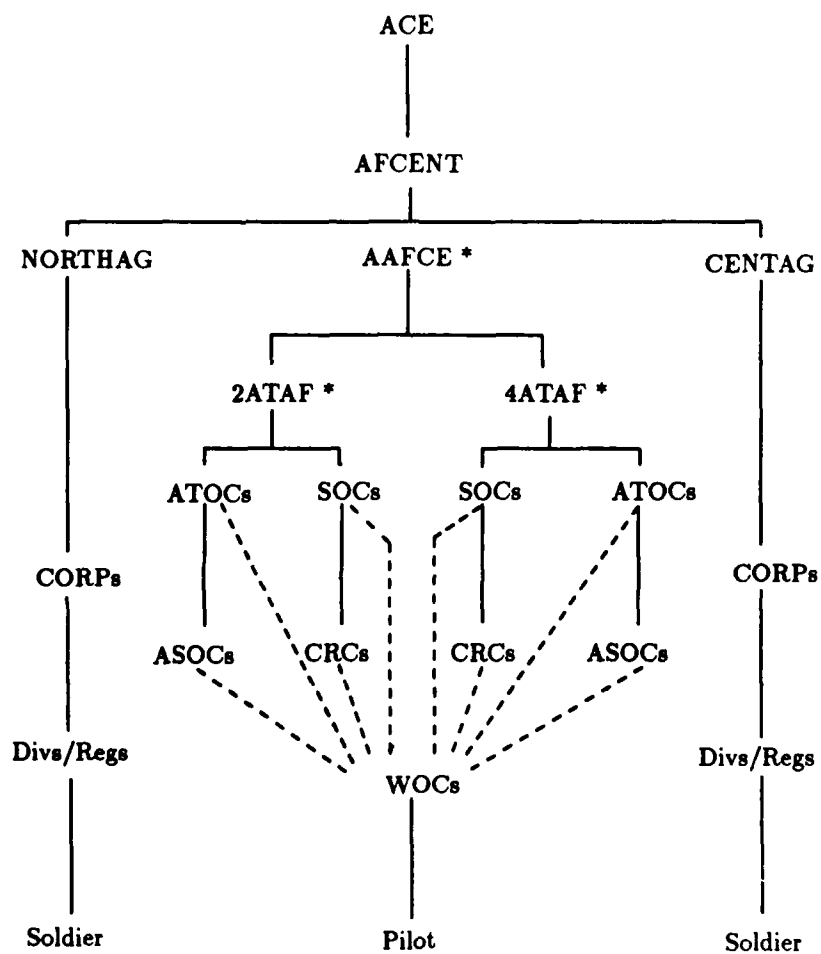
¹In 1984, the wargaming personnel and functions of HQ AU Data Automation were transferred to the newly formed Air Force Wargaming Center under The Center for Aerospace Development, Research, and Education

is always composed of 8 to 15 student players while their opponent, the red team, is usually composed of Air War College faculty members trained in Warsaw Pact doctrine.

Basically, the exercise is played at two different levels of command structure contained in NATO's Central European Region. First, the Allied Air Forces Central Europe (AAFCE) level is played with each student in the seminar assuming some role within the AAFCE organizational structure, either as the AAFCE Commander or part of his staff. At this level, the players formulate an overall air strategy based on Higher Headquarters guidance and decide on how to best position forces and logistics to support that strategy. This requires the players to make decisions based on tradeoffs between obtaining air superiority while still providing support for allied land unit operations. A few of the typical player decisions would be where to beddown aircraft to obtain maximum sortie utilization and dealing with the logistical problems associated with supporting aircraft operations. Figure 1 depicts the complete NATO Central European Region Command and Control Structure with the levels played by the Theater War Exercise annotated.

Next, after the strategies and decisions for the AAFCE level have been accomplished, the Allied Tactical Air Forces (ATAF) level is played. At this level, the players (students) implement the air directives and objectives they made at the AAFCE level while ensuring optimal use of their limited resources. This involves selecting appropriate targets, i.e., enemy land units and airbases, and developing force packages (air missions) to strike those targets. Figure 2 shows the organizational chart for the various roles the players will assume at the AAFCE level while Figure 3 shows the player roles for the ATAF level.

The Theater War Exercise requires the players to make decisions typical of those that an air component commander and staff (AAFCE or ATAF) would make during an actual war. These decisions are fed into the TWX air and land battle simulation programs which model the employment of the airpower strategy, doctrine, and warfighting principles inherent in those decisions. Once the simulation is run, the players receive their battle results, logistic status, weather forecast, and new



* Denotes TWX Player Roles

Figure 1. Central European Command and Control Structure

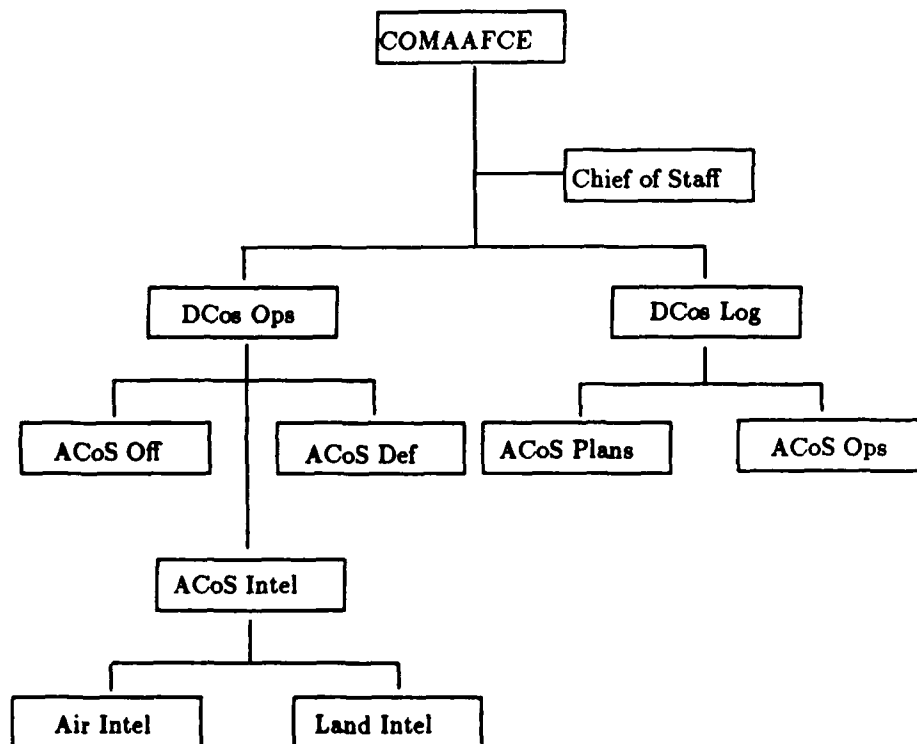


Figure 2. AAFCE Player Roles

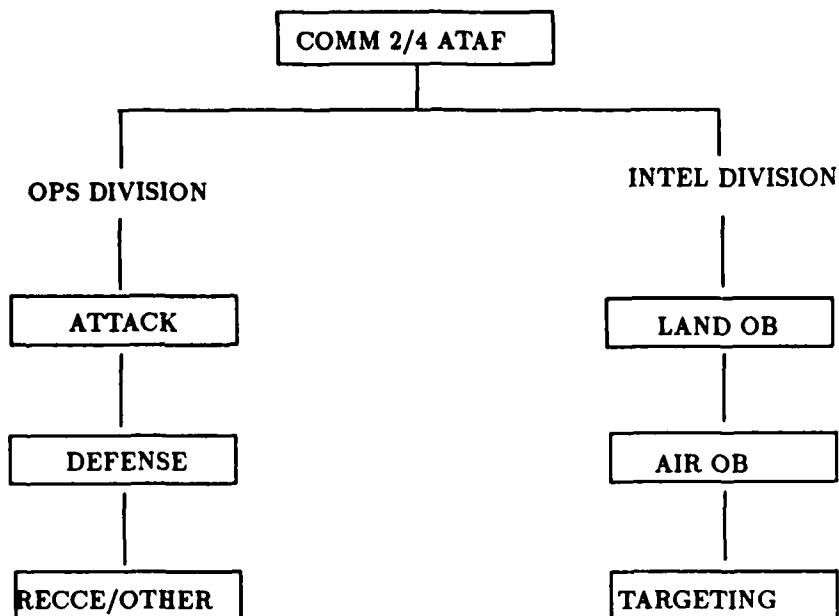


Figure 3. ATAF Player Roles

air/land orders of battle in a computer printout format. Using these results along with enemy intelligence estimates, the players revise their strategy as necessary and prepare the next day's battle plan. After the fifth and final day of the exercise, the students are critiqued on their performance by the course instructors, based on their overall exercise results.

The main purpose of this exercise is not to determine if a particular seminar won the war, but to allow the players to gain an understanding and appreciation of the difficult decisions and pressures associated with high level command positions. The data and simulation algorithms used within the exercise are unclassified but reflect reality wherever possible. However, since the air and land battle simulation algorithms have never been formally validated, the results of any particular exercise cannot be perceived as a true indication of how a real world European theater conflict would proceed based on the player's decisions. The Theater War Exercise does, however, provide a valuable educational insight into the application of airpower employment strategy and doctrine [3].

1.2 Problem Statement

The purpose of this thesis effort was to enhance the TWX system by incorporating modern software and hardware technology. This was accomplished by designing and implementing a new Theater War Exercise database system which will support a more friendly TWX user interface and provide more flexibility for maintaining the exercise data. Also, in order to provide better exercise portability, the TWX air and land simulation software, along with the new database system, was transferred to a networked, microcomputer configuration.

1.3 Justification

As noted, the Theater War Exercise is a mature computer system which has changed little since its initial conception. Although, the students gain valuable experience from participating in an exercise such as TWX, even greater benefit could be derived if the system was enhanced to provide an environment more in tune with today's computer literate user. The following sections address the reasons for enhancing the TWX exercise along with the objectives of this project.

1.3.1 Current System Constraints. In its present configuration, the exercise is a large, Fortran IV program which runs on a 1970's vintage, Honeywell H6000 mainframe computer. All user inputs are made on slow, hard copy printers which severely limit the interaction by game controllers and exercise participants. Because of these limitations, players cannot receive on-line help from the computer and have no way of correcting data input errors. For example, in order to input ATAF air missions, the users must follow a very rigid format which does not allow for any keystroke or mental mistakes. Once the player information is typed in, it is very difficult to change (often requiring game controller intervention).

This unfriendly user interface tends to frustrate the players because they must learn a complicated computer syntax rather than concentrating on playing the exercise. Also, due to processing constraints built into the software, players cannot process logistic input concurrently with ATAF

mission input. This creates a lot of pressure on the players performing logistic functions to meet unrealistic deadlines to complete their data transactions so that ATAF mission input can be accomplished. These factors, plus simulation constraints such as fixed number of airbases, aircraft, etc, makes the exercise inflexible and difficult to enhance and maintain. For example, the current system is limited to a maximum of 80 airbases, 50 different aircraft types, and 10 different aircraft munition types per side. Although, these figures seemed quite adequate when the game was first developed, as the game evolved, these constraints limited the realism associated with the exercise by restricting the scope of theater operations, the addition of new aircraft types, and the types of munition loads associated with various aircraft missions.

As stated previously, TWX is also played each year at the Canadian Forces Command and Staff College in Canada and the Royal Air Force Staff College in England. This off-site processing currently requires a compatible Honeywell H6000 computer facility to be leased at each foreign location and Air Force Wargaming personnel to setup and conduct each exercise. Because of the expense involved with leasing a H6000 computer and the travel of Wargaming personnel, the exercise is only conducted at these foreign sites annually.

1.3.2 Project Objectives. In order to alleviate these exercise constraints, this project focused on two major objectives. First, the exercise user interface was modified to incorporate good software engineering design principles. Here, a commercial database management system (DBMS) and 4th generation language (4GL) application development tool were used to design user interface screens which incorporate help facilities, on-line editing, and real-time data validation. This will remove a tremendous burden from the seminar players and will allow them to concentrate on playing the exercise rather than mastering the computer environment. Also, by using a DBMS to perform the data manipulation for the exercise, a tremendous amount of flexibility for creating and maintaining the multiple TWX seminar databases was added. This will make the game controller's job easier while allowing ongoing enhancements to be quickly implemented.

	Micro-TWX	Honeywell TWX
Data Management	Ingres DBMS	Fortran managed files
-add data fields	done easily	change programs
-data limits	limited by simulation	limited by simulation and input programs
-data changes/queries	ad hoc on all data	limited by fortran utilities
	concurrent with users	users must log off
User Interface	user-friendly	user-hostile
-General	screen or printer output	printer only
	concurrent AAFCE/ATAF	serial AAFCE/ATAF
	screens easy to change	input programs hard to change
	bad entries correctable	retype entire line
	99 airbases/side	80 airbases/side
	80 aircraft/side	50 aircraft/side
	20 munition types/side	10 munition types/side
-AAFCE	immediate feedback on changes	run a report
	multiple inputs per screen	one input per line
-ATAF	packages by target	packages by aircraft
	edit & delete missions	no edit or delete
	reports	no reports
Enhancements?	unbounded	difficult

Figure 4. Comparison of Micro-TWX and Honeywell TWX

Finally, by re-hosting the exercise to a smaller and more compact computer architecture, better system portability was obtained which will allow the game to be conducted at remote sites, i.e., Canada and England, without the need for mainframe computers or the support of Air Force Wargaming Center personnel. This was accomplished by targeting the new TWX system to a combination of standard IBM PC/XT compatible microcomputers and a Digital Equipment Corporation MicroVAX II. This setup will provide an environment which is powerful enough to support up to 8 simultaneous users configured such that up to 4 separate exercises could be running concurrently. By accomplishing these goals, a user friendly, microcomputer based Theater War Exercise can be played at any site using relatively inexpensive hardware and software as compared to the mainframe Honeywell TWX system. Figure 4 shows a comparison of the new Micro-TWX system versus the old Honeywell TWX system.

1.4 Assumptions

The development of the Theater War Exercise microcomputer system proceeded based on the following assumptions:

1. The Air Force Wargaming Center was satisfied with the operation of the current air and land simulation software (excluding the user interface and software processing constraints) and did not desire any further enhancements to this software.
2. The new TWX microcomputer version will be validated if it produces output results consistent with the current TWX system given the same input parameters. Final system validation will be based on representatives of the Air Force Wargaming Center evaluating and accepting the results.

1.5 Scope

This thesis project developed the database system and associated software necessary to reconfigure the TWX system from the Honeywell H6000 computer to a Digital Equipment Corporation Micro-Vax II microcomputer / Zenith Z-158 PC combination. This effort included transferring the TWX air battle simulation software and documenting the development with a TWX user's manual and systems integrator's handbook (maintenance and installation procedures). However, this project did not make any enhancements to the TWX software outside of those necessary to incorporate a database management system and accomplish the air simulation conversion processes while incorporating a new mission targeting scheme.

A separate thesis project [8] handled the design, implementation, and documentation of the software necessary to implement the TWX microcomputer user interface.

1.6 Development Approach

This thesis project was accomplished in three distinct phases. First, the database design and implementation was accomplished so the new user interface software could be rapidly implemented and tested. Second, the air battle simulation software was modified and transferred to run on the new microcomputer configuration.

Finally, the combined software from the air battle simulation along with the database and user interface software were fully integrated and tested. At this time, the necessary TWX documentation was also completed.

The following list outlines the steps used for accomplishing this project:

1. Database Design and Implementation

- (a) An Entity-Relationship (ER) diagram was developed using the current TWX database and reporting requirements as a guide.
- (b) The TWX database management system requirements were determined based on the ER model, i.e., what capabilities were necessary for a commercial database system to meet the TWX processing requirements.
- (c) Using the techniques outlined in Chapter II, a preliminary evaluation of the available commercial database management systems was conducted. However, this process was not completed due to the Air Wargaming Center making the final DBMS selection before a comprehensive evaluation of the final DBMS candidates could be accomplished.
- (d) Using the database design techniques outlined in Chapter III, the TWX database was designed and implemented using a commercial DBMS.
- (e) An application program which will allow the exercise controller to maintain the various seminar databases and monitor each exercise was developed.

- (f) Finally, the database maintenance and system administrator manuals were written in draft form.

2. Air Battle Simulation Software Conversion

- (a) The user interface and land battle simulation programs were separated from the air battle simulation program.
- (b) The sections within the air battle simulation software which required interaction with the new database system were modified.
- (c) The air battle simulation was modified to accept a new mission targeting scheme.
- (d) The remaining air battle simulation routines were transferred and the system tested.
- (e) Finally, the air battle simulation software maintenance manual was updated in draft form.

3. System Integration and Testing

- (a) The final user interface software was integrated with the database management system and tested.
- (b) The air battle simulation software, database system, and user interface were combined to form an integrated system.
- (c) The new TWX system was validated against the old TWX system.

1.7 Material and Equipment

The Air Force Wargaming Center provided one Digital Equipment Corporation Micro-Vax II and three Zenith Z-158 microcomputers along with the commercial DBMS and associated software required for this thesis project. They also provided one listing of the current TWX source code and one complete set of documentation (operators, user, and maintenance manuals).

1.8 Sequence of Presentation

The remainder of this thesis is divided into five separate chapters. Chapter II describes the rationale for incorporating a commercial database management system into the new TWX implementation along with the advantages and disadvantages of the various database models under consideration. This chapter also addresses the TWX DBMS requirements, the DBMS evaluation and selection process and the TWX computer hardware selection process. Chapter III discusses the relational database design method used for this project and how it met the TWX system requirements.

Chapter IV describes the air battle simulation software conversion issues along with the air simulation software validation. Finally, chapter V discusses the process of developing the Micro-TWX controller application and chapter VI contains the conclusion and recommendations for further work.

II. Software and Hardware Selection Issues

This chapter addresses the issues concerned with selecting the commercial DBMS software and computer hardware for the Theater War Exercise conversion. First, the rationale for replacing the current application database system with a commercial DBMS is discussed along with an overview of the database models which were considered for this project. Next, the DBMS evaluation methods used for this project are reviewed and the Micro-TWX DBMS requirements outlined. Finally, the Micro-TWX hardware selection is discussed pointing out what capabilities were necessary for this implementation.

2.1 Rationale for Using a DBMS

As previously noted, the current TWX system was developed in 1977, prior to the advent of economical and fully functional database management systems. Because of this, the database function within the current TWX system is handled by a collection of independent application programs which access multiple data files via the Honeywell operating system. This type of design approach reflects an emphasis on physical implementation rather than logical data organization and is very hardware and operating system dependent.

According to Howe [6], this type of database design approach has several faults. First, using independent application programs for a database system can create a large amount of data redundancy due to the need to duplicate data items among several different application files. This duplication can cause a decrease in system performance since more overhead is associated with maintaining the database in a consistent state, i.e., ensuring all duplicate data contains the same values. Also, since these application programs must rely on knowing the physical location of the data to access it, the data structures within these files cannot be easily modified without requiring changes to each application program which must access the file. This fact significantly increases

the amount of work for the maintenance programmer any time modifications or enhancements are made to the system.

These factors are especially true within the existing TWX database system where data items for logically related entities are contained in separate files and often duplicated. One particular example concerns the way the information for targets, airbases and land units is handled. Although airbases and land units are both targets within the exercise, the target information pertaining to these entities is maintained separately from the airbase and land unit information. This requires multiple file accesses each time a land unit or airbase is added or modified to ensure the target file reflects the new changes. Also, if the data structures for any of these files are changed, each application program which accesses these files must be modified.

Another reason for not using the current TWX file system is that the specialized database application programs would have had to be rewritten for the new microcomputer architecture. This would have required all the specialized Input/Output code within each routine to be modified along with any hardware dependent file management system calls. This course of action would have entailed a major software rewrite since the file structure for the proposed microcomputer hardware is significantly different from the current Honeywell system.

Because of the problems associated with trying to rewrite the current system, it was more advantageous to incorporate a vendor supplied DBMS in place of the current application database system. This was especially true since the same data duplication and maintenance problems would still exist in the new version if the current system was used.

The basic function of a database management system is to provide the necessary software to define a database by specifying its logical data structure and to perform all modifications upon the database [4]. A DBMS will also perform all data additions, deletions, and retrievals while providing data security, integrity checking, and minimizing data redundancy [6]. By using a commercial

DBMS, the new TWX system was developed around a centralized database scheme which was easier to design and will be much easier to maintain and enhance.

Another benefit of incorporating a DBMS into the TWX system was that it provided a 4th generation language (4GL) along with an automated forms management system. A 4th generation language is a highly specialized programming language which allows an application developer to quickly develop functional applications around a menu based system. These tools were used to develop the new user (player) interface using a rapid prototyping design approach [8].

To incorporate these DBMS features into the new TWX system, the current user interface software was removed from the system and replaced by 4th generation applications software. Also, the present file I/O calls within the actual simulation programs were replaced by embedded database statements which directly access the database via the DBMS. Although this involved a major software coding effort, it was easier than trying to convert the old Fortran IV application routines into a functional and user friendly product.

2.2 Database Models

The commercial database management systems considered for the TWX project were developed around either the relational or the extended-network data models. A data model is basically used by an application developer to describe the logical structure of a database (or scheme). It consists of the conceptual tools for describing data, their relationships, semantics, and constraints [5].

The data model also provides a logical view of the types of data that exist and the nature of their interrelationships [4]. Each type of model has certain strengths and weaknesses and differs primarily in terms of how it can represent relationships among data types. The following sections briefly describe the relational and extended-network models and how they matched up with the TWX database requirements.

Airbase Table

ab-id	ab-side	ab-name	...
50	blue	Ramstein	
63	blue	Bracknell	
75	red	SAFP AD	
52	red	EGAF AD	

Aircraft Table

ac-name	ac-side	sortie-rate	...
F4G	blue	5	
F111E	blue	3	
B52G	blue	2	
MIG17	red	6	

Aircraft-on-Airbase Table

ab-id	ab-side	ac-name	quantity
50	blue	F4G	15
50	blue	B52G	5
63	blue	F111E	14
75	red	MIG17	30
52	red	MIG17	20

Figure 5. Sample data as depicted by the Relational model

2.2.1 Relational Model. The relational data model was developed by E.F. Codd and is essentially a computer based implementation of the mathematical notion of a relation [7]. Basically, the relational model consists of a collection of tables called relations which provide a tabular display of the data. Each table is divided into columns which are called attributes and rows which are called tuples. This feature provides a conceptual view of the database which is unencumbered with details of the underlying implementation, and because of its tabular nature, the model is easily understood by users and is easy to implement. Figure 5 shows an example of some relational tables using sample TWX exercise data.

The relational model also provides formal criteria for designing logical data structures called normal form relations [5]. These criteria (normalization techniques) are used to achieve a database design which minimizes data redundancy while maintaining the ability to efficiently test data integrity. The relational model also uses a non-procedural query language which describes what the user wants rather than how to retrieve it [7]. This feature is important because a user or application developer does not need to understand how the underlying database is constructed in order to retrieve or update information.

However, according to Bonczek [4], the relational model does have two distinct disadvantages. First, it must use some data redundancy features to represent relationships between data. These redundancy features add to the amount of physical data stored within the database and could have a negative impact on the TWX microcomputer implementation if the amount of secondary storage available is a problem. Second, although the relational model has data integrity features, most commercial implementations based on the relational model do not. Because of the lack of integrity features, the designer must use traditional application programs or fourth generation application tools to ensure the referential integrity of the data [4]. This fact is important because data integrity within the TWX system is crucial since the exercise players will be under severe time constraints to complete their data input thus, prone to making input errors.

2.2.2 Extended-Network Model. The extended-network model (commonly called post-relational) is the newest approach in data modeling techniques. The extended-network model differs from the relational model in that data is represented by collections of records whose relationships are represented by links or pointers instead of tables [4]. Each record belongs to an owner-coupled set in which an owner record from one object class is associated with any number of member objects from another object class [5]. This allows complex data relationships to be directly represented versus the relational model where these relationships have to be forged using duplicate data fields

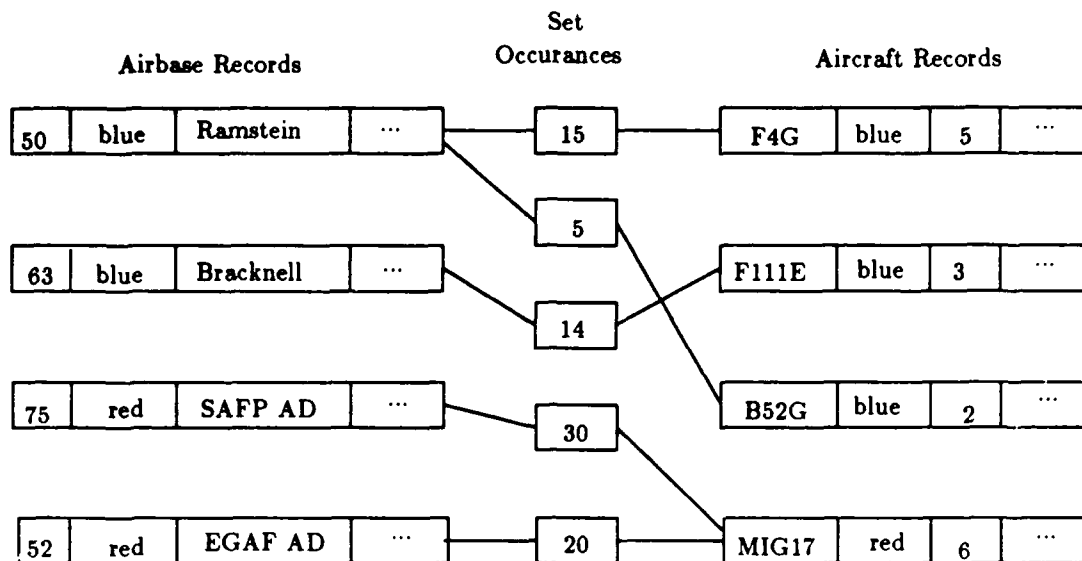


Figure 6. Sample data as depicted by the Extended-Network model

[4]. Figure 6 shows the sample TWX data used in figure 5 represented in an Extended-Network model.

Although the relational model is easier for users and designers to understand, i.e., tables versus owner coupled-sets, the extended-network model has several advantages over the relational model. First, it provides data integrity automatically without the need for external application programs or fourth generation tools to enforce integrity [4]. Secondly, since it uses pointers to keep track of relationships, it does not contain duplicate data like the relational model. However, there is some minimal space overhead associated with the pointers.

Finally, it is easier for the developer to model real world occurrences since complex data relationships can be represented with minimum effort [4]. For example, some of the data relationships within the TWX database are very complex and diverse. One ternary relationship which exists in the TWX database (used to compute the destructive power of an air mission) is made up of entities consisting of aircraft, weather, and munition packages. This type of relationship can be difficult

to implement using the relational model because a separate relation containing the keys for each entity must be created along with the necessary data integrity software. The extended-network model can handle this type of relationship in a much easier fashion using pointers.

However, like the relational model, the extended-network model also has some drawbacks. First, because it uses pointers to logically connect the data and enforce integrity, it can be very difficult for the user and developer to understand the database scheme. Secondly, the extended-network model uses a procedural query language which forces the user to understand the underlying logical database implementation in order to retrieve information. Since the TWX database is very large and the exercise requires very complicated reports, it could be difficult for a user (i.e., game controller) to make "ad hoc" queries or dynamic database updates.

2.3 DBMS Evaluation Methods

Evaluating database management systems can be a complex and time consuming task. According to a software newsletter, there are over 20 commercially available DBMS products for microcomputer applications and the differences between them can be very subtle [10]. Without some type of formal evaluation process an adequate and fair comparison of each system to each other and the TWX database requirements was impossible. For the TWX project, the best traits of the following DBMS evaluation methods were used to narrow the field down to three contenders.

2.3.1 BIS Applied Systems, Ltd Approach. BIS Applied Systems, Ltd is a software consulting firm who has published a database management system evaluation approach. Their approach consists of a series of steps which are carried out for each DBMS evaluated. First, a paper evaluation is conducted which matches the capabilities of the particular system to the requirements of the proposed database [12]. Its main purpose is to allow the evaluator to become familiar with all the database management systems available and weed out those which do not meet certain standards.

Next, a theoretical benchmark is accomplished for each system. This simulates the processing requirements of a sample application using the published performance characteristics for the system [12]. Its purpose is to further weed out systems which do not meet certain well defined criteria.

Finally a physical benchmark and live system evaluation are conducted [12]. Here, the objective is to test each database package under actual system conditions. Since this can be expensive (demonstration versions are not free) it was impossible to actually perform a live system evaluation for each DBMS considered for the TWX project. However, this step can provide the most reliable data as to the capabilities for each system evaluated [12].

Using all the steps within the BIS Applied System method for the TWX DBMS evaluation was impossible. Granted, the paper evaluation applied and was extensively used, but running a physical and live system evaluation for each DBMS was impossible due to lack of funds. Also, there are professional software rating services which have benchmarked most of the top database management system products [10] and these ratings were used to reinforce the paper evaluations.

2.3.2 CLEC System. The Comprehensive List of Evaluation Criteria (CLEC) system was developed by Lehot and Kaisler as a guide to evaluating competing data base management systems [9]. The CLEC system is composed of a comprehensive list of evaluation criteria which is divided into classes of features and capabilities desirable within a database management system. For example, one class may concern security issues and another backup and recovery capabilities.

This system uses a methodology for assigning weights to individual classes and aggregates grades assigned as judgements to candidate database management systems. This provides a quantitative figure-of-merit for each system evaluated which can then be used to rank them in numerical order. This method was directly applied to the TWX DBMS evaluation problem. By analyzing the necessary TWX database requirements and applying this method the available commercial DBMS packages were evaluated and ranked.

2.4 Micro-TWX DBMS Requirements

In order to ensure a fully functional product for the TWX microcomputer implementation it was crucial that the right DBMS was selected. Based on the preliminary TWX MicroVAX/Z-158 design, the following capabilities were required for any potential DBMS candidate to be considered for the Micro-TWX project:

1. Operate on both a MicroVAX II and a Z-158 microcomputer in a Local Area Network (LAN) or direct line configuration. This feature allowed the TWX system to utilize the processing power of the Z-158s which reduced the processing load on the MicroVAX, thus, supporting more simultaneous users.
2. Provide a powerful 4th generation applications development tool. This feature was necessary to support an applications development environment which allowed forms production, windowing, help facilities, and sophisticated screen manipulation features using a mouse or programmable function keys.
3. Powerful data manipulation language. Had to perform complicated queries quickly and efficiently. Necessary to provide real-time data validation and help facilities.
4. Powerful report writer. Essential to provide the complicated reports required for this project. Without this feature reports would have had to be written in a high order language, thus increasing workload, program complexity and maintenance requirements. Also, a good report writer had the capability to support "ad-hoc" reporting requirements.
5. Have a host language interface. This capability was necessary in case the report writer could not support some of the complex TWX reporting requirements. Also, this capability was essential for converting the air and land battle simulation programs to a DBMS based system.

6. Provide data integrity checking. Crucial. In an environment where the players will most likely be inexperienced computer users, ensuring the data is not corrupted is of the utmost importance. A DBMS which performs range checking, table lookups, calculated fields, and referential integrity checking would greatly simplify the implementation by reducing the amount of programmer developed software.
7. Transaction logging capability. This capability was not essential but would provide the ability to recover from system crashes or even roll back (replay) a section of the simulation.
8. Fast data manipulation capability. Must be able to store and retrieve records, while performing data validation, in three seconds or less. An acceptable user response time was necessary to make the TWX application functional.

After reviewing numerous vendor products, the systems available fell into two different categories; extended file management systems and true database management systems.

Extended file management systems typically only provide a subset of the capabilities of a true database management system. For example, a file management management system will not usually provide a general data manipulation language, query optimizer, or a choice of data storage structures, e.g., hashed and indexed. Neither will they provide a concurrency control scheme, transactional logging or referential integrity checking capabilities. Although file management systems generally cost less, using one for the TWX microcomputer implementation would have impacted both the level of sophistication in the user interface and the integrity of the database.

2.5 Micro-TWX DBMS Selection

Of the 12 commercial systems evaluated for this project, there were only three "true" database management systems which met most of the previously stated TWX requirements; MDBS III, an extended-network DBMS, Oracle, a relational DBMS, and Ingres, also a relational DBMS. Using a

combination of the DBMS evaluation methods discussed previously, the strengths and weaknesses of each system were compared to each other.

Overall, the best package with regard to the database function for the Micro-TWX implementation was MDBS III. MDBS III, using the extended-network model, was better suited to handle the intricate data relationships required for the Micro-TWX database. Although the relational systems Oracle and Ingres can also handle these relationships, MDBS III outperforms both of them because it would not have required any data duplication and could access data quicker because it does not have to perform multi-way table joins like the relational systems.

However, at the time of this evaluation, the MDBS III PC version did not support a 4th generation applications development tool or a forms management generator. Without these features, the processing power of the Z-158 could not be utilized unless the user interface application software was developed in a high-order-language such as Pascal. Since a 4GL and forms management system were needed to rapidly develop the user interface, MDBS III was eliminated from consideration and the selection process focused on the two remaining database systems, Oracle and Ingres.

Both the Oracle and Ingres database management systems were proven performers and had a powerful 4th generation application development tool which included a forms development package. Either system provided all the capabilities necessary to develop a professional Micro-TWX user interface incorporating on-line help, windowing, and referential integrity checking features. Also, both systems had a powerful report writer which was capable of producing the complicated reports required for this project.

However, before the formal evaluation was completed, the Air Force Wargaming Center selected Ingres for this project due to one overriding factor. Since, the Center is currently using Ingres for other in-house database applications and already has a trained support staff, they selected Ingres to preclude unnecessary training. This factor, plus significant cost reduction incentives, made

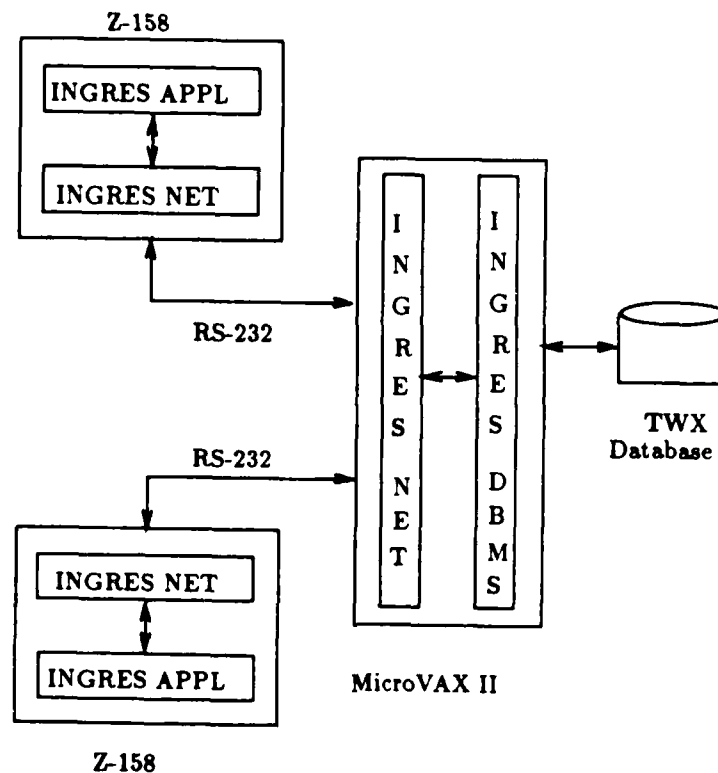


Figure 7. Micro-TWX DBMS Configuration

Ingres the logical choice. Figure 7 depicts the proposed Ingres DBMS setup for the Micro-TWX project.

2.6 Hardware Selection

In selecting the hardware for the new Micro-TWX system there were several factors to consider. First, a multi-user system or a local-area-network (LAN) configuration was needed to support a minimum of three users, i.e., blue players, red players, and a game controller. Second, a powerful central processor was required to run the actual game simulations in an acceptable amount of time. In most cases, the actual simulation portion is run at night so turnaround time is not that critical. However, the exercise schedule currently requires a four hour maximum turnaround for the final day of the exercise.

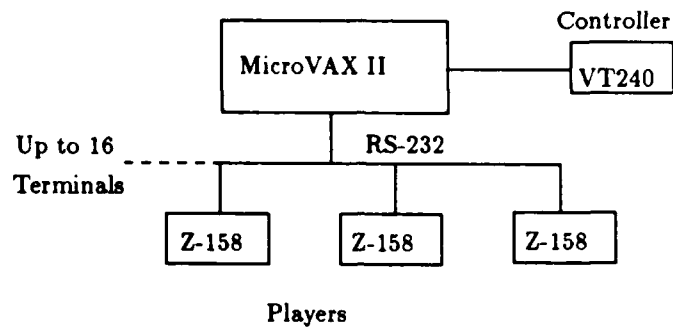


Figure 8. Micro-TWX Hardware Configuration

Finally, a large amount of secondary disk storage was required to store the TWX database and simulation programs. Depending on the number of seminars the system must support, anywhere from 20 to 100 MB of storage will be needed.

To solve these requirements and still meet the project goal of portability a MicroVAX II, Z-158 PC/XT combination as depicted in Figure 8 was selected. By networking the Z-158s and MicroVAX together, the MicroVAX will perform as a central file server for concurrent exercises, with the Z-158s running the user interface software. Also, the MicroVAX, using the MicroVMS operating system, has the capability to run batch jobs which will allow large report jobs and the game simulations to run in the background, thus, freeing up the individual player's terminal for other processing requirements.

III. Micro-TWX Database Design and Implementation

This chapter addresses the issues concerned with the design and implementation of the Micro-TWX relational database system. This database will serve as a repository for all the data necessary to conduct the Theater War Exercise. This includes initialization data, player inputs, and simulation constants used throughout the game. First, a brief overview of the bottom-up and top-down design methodologies is presented, followed by an description of the integrated design approach used for this project. This includes a description of the extended Entity-Relationship (E-R) modeling process and its mapping to a relational system. Finally, the actual database implementation using the Ingres database management system is presented, discussing some tradeoffs concerned with query processing for the user interface and the options for supporting multiple exercises.

3.1 Design Approach

The database design approach used for the TWX project is a combination of top-down and bottom-up methodologies. This integrated approach was selected because both the top-down and bottom-up relational design methods have some unique drawbacks when used by themselves.

For example, when using the traditional bottom-up design method, the designer must perform a series of steps where he identifies all the required database attributes and their data dependencies. He then must perform a normalization process which decomposes the attributes into relations.

To elaborate, an attribute is the most basic piece of information within a database system and is directly associated to a column in a relation (i.e., ab-id is an attribute in the airbase relation). Data dependencies are used by the designer as a means to imply that values of various attributes within a relation can be dependent upon one another. Data dependencies can consist of both functional and multivalued dependencies. A functional dependency (FD) $X \rightarrow Y$ states that for each X -value there is a unique Y -value associated with it. For example, using the attributes "ab-id" and "ab-name" from the airbase relation, the FD $\text{ab-id} \rightarrow \text{ab-name}$ states that for each

airbase identifier there exists a unique airbase name. On the other hand, a multivalued dependency (MVD) $A \twoheadrightarrow B$ states that attribute set A multi-determines attribute set B. For example, given the relation with attributes (A,B,C), if the tuples (a, b', c') and (a, b, c) exist, then the tuples (a, b, c') and (a, b', c) must also exist [7]. The normalization process involves analyzing the various dependencies between the entity attributes and decomposing them into relations using a normalization algorithm in order to produce a database which is in a normal form. These various steps are necessary in order to produce an optimal database design which minimizes data redundancy while still providing a means to enforce data integrity.

Although the bottom-up method produces optimal database designs, it can be an overwhelming task when applied to large databases such as TWX. On the other hand, the strictly top-down design approach using semantic models to directly implement a functional database system is still in a theoretical stage [13]. Here, researchers are trying to directly implement an efficient database environment by using a high level abstract model such as an Entity-Relationship diagram (i.e., the designer would only have to design the E-R diagram and not the underlying system). Because of the constraints associated with both of these design approaches, a more practical top-down, bottom-up combination approach is used by most professional database designers [13].

The specific logical design approach used for this project is described by Teorey, et al. [13] and Korth and Silberschatz [7]. With this approach, the top-down method of using conceptual design tools (semantic models) such as entity-relationship (E-R) diagrams has been combined with the traditional data dependency and normalization (bottom-up) approach. The basic steps of this methodology are summarized as follows [13]:

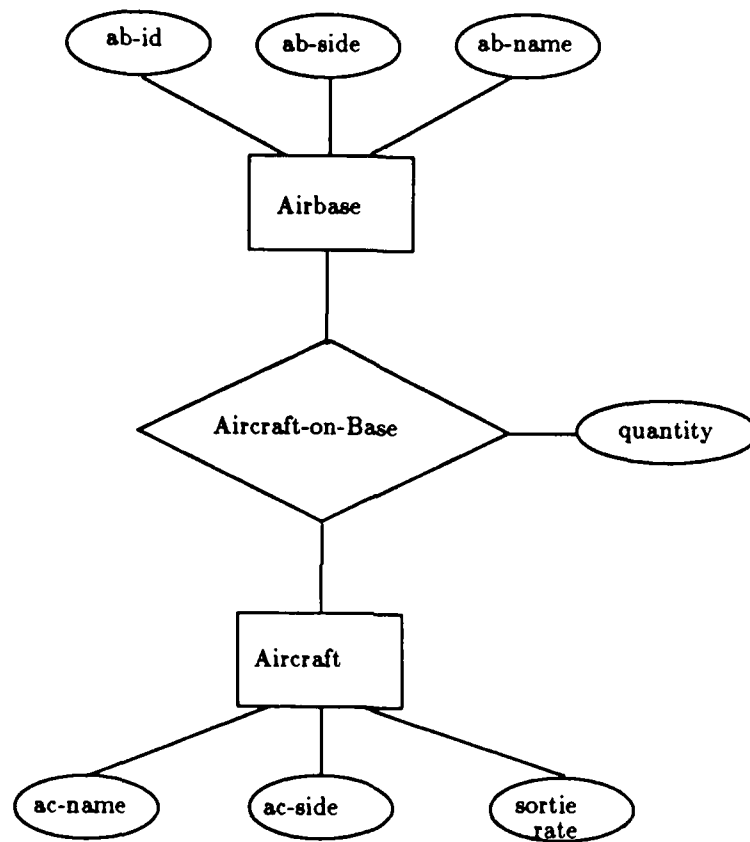
1. Analyze and model the data requirements using an extended E-R diagram and identify primary keys for each entity.
2. Transform each entity and relation within the E-R diagram into candidate relations (tables) and remove redundant relations.

3. Identify any non-trivial functional and multivalued dependencies for each relation. Next, normalize the relations to the desired normal form using standard normalization techniques. Again, remove any redundant normalized relations which do not affect dependency preservation.

This design method has several advantages. First, by using the concept of abstraction, via E-R diagrams, the number of entities or objects to keep track can be an order of magnitude less than the number of data elements in the database [13]. This, in-turn, simplifies the database design process by reducing the number of data dependencies that need to be analyzed which allows the designer to focus on the logical scheme rather than the physical implementation [13]. One particularly nice feature of this process is that if the E-R diagram is designed correctly, the multivalued dependencies are represented by the relationships between entities. This reduces much of the design effort in the data dependency identification phase. Also, data integrity is preserved through the normalization of the candidate relations derived from the transformation of the E-R model.

3.1.1 Entity-Relationship Process. As stated, the first step in this design process was to develop an E-R diagram which facilitated the database design by specifying an enterprise scheme. This scheme represented the overall logical structure of the database and provided an intuitive view of how the database would function.

The Entity-Relationship model was originally developed by Chen and consists of entities (object sets), relationships between entities, and attributes which describe either an entity or relationship [7]. For example, within the TWX E-R diagram, "airbases" and "aircraft" would be entities with attributes "ab-id, ab-side, ab-name" and "ac-name, ac-side, sortie-rate" respectively. A relationship such as "aircraft-on-base" with attribute "quantity" would be an association of the airbase and aircraft entities describing the type and number of aircraft on each airbase. Figure 9 depicts the associated E-R diagram for this example along with the each entity's functional dependencies.



Functional Dependencies
ab-id ab-side \rightarrow ab-name
ac-name ac-side \rightarrow sortie-rate
ab-id ab-side ac-name ac-side \rightarrow quantity

Figure 9. Partial Micro-TWX E-R Diagram

In order to adequately represent the complex relationships within the TWX database an extended version of the E-R diagram which evolved from the original E-R model was used. The extended E-R diagram provided mapping cardinalities which defined the number of entities to which another entity can be associated through a relationship [7]. For example, there are one-to-one constraints which means that an entity X can only be associated with one entity Z and vice versa within a relationship. Also, there are one-to-many, many-to-one, and many-to-many mapping cardinalities which cover any real world situation which needs to be modeled. Figure 10 depicts an example of a many-to-one and many-to-many mapping cardinality using sample TWX data. The many-to-one constraint is represented within the land-unit to corps relationship "is-in" by the directed line going into the corps entity. This implies that each land-unit can only belong to one corps while a corps can contain many land-units. However, the many-to-many cardinality is represented within the land-unit to vehicles relationship "has vehicles" by a non-directed line. This implies that a land-unit can have many different types of vehicles and a vehicle type can be associated with many different land-units.

Another abstraction the extended E-R model supports is that of generalization and specialization, i.e., the "IS-A" construct [7]. Generalization means that several lower level entity sets can be combined to form a higher level entity set and specialization means lower level entity sets are derived from a higher level entity set [7]. This capability allows the database designer to assign unique attributes and relationships to the "IS-A'ed" entities while they share the general entity's attributes and relationships.

For example, within the TWX E-R diagram there were several instances of the "IS-A" construct. One such instance is a specialization of the "air-mission" entity where the entity is broken out into different specialized types of air missions, i.e., close air support (CAS), reconnaissance (REC), offensive counter air (OCA), and interdiction (IND). Without the "IS-A" construct this real world situation would have been very difficult to model because each mission type can be

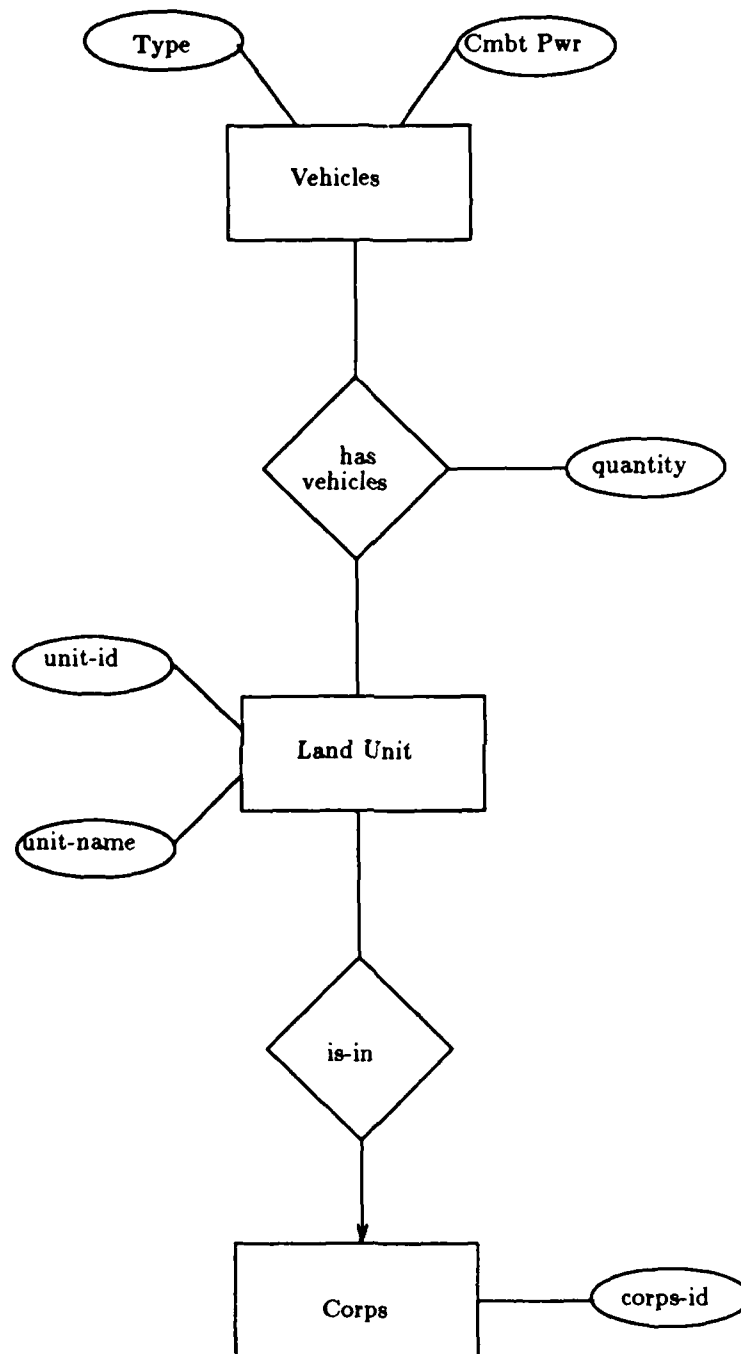


Figure 10. Sample E-R Diagram Showing Mapping Cardinalities

associated with only certain types of targets. Figure 11 depicts a sample E-R diagram for the previous example. Also, note the many-to-one mapping constraint between the air mission entities and target entities as denoted by the directed line into the targets. This implies that each unique air mission can have at most one target and that a target can have many air missions directed against it.

For the TWX design, the current application file system along with the documentation for the air and land simulation programs were used as a guide to construct the E-R diagram [2]. This involved analyzing each of the 15 separate data files to determine what entities and relationships existed within the exercise. Also, personal interviews with individuals familiar with the exercise logic [11] were necessary to determine the mapping constraints between entities. Finally, an analysis of the current game input and reporting requirements as depicted by the Theater War Exercise Users Manual [3] was conducted to ensure the necessary data was represented within diagram. The final E-R diagram for the Micro-TWX database consisted of over 30 entities (e.g. airbases, aircraft, land units, etc) and over 40 different relationships between these entities.

3.1.2 E-R to Relation Conversion. After the E-R diagram was completed the entities and relationships were transformed into candidate relations (tables) according to step 2 of the logical design process. Special types of transformation rules applied to each E-R construct, i.e., one-to-many relationship, "IS-A", etc. [13]. For example, a many-to-many relationship between two separate entities requires a relation (table) with primary keys from both entities along with any unique attributes associated with that relationship. By following the special rules outlined by Teorey, et al., the E-R diagram to candidate relation transformation was a straightforward process. Also, there were several redundant tables which were removed during this step. For example, by moving the "mission-type" attribute from the air-mission relation to the air-mission-ac relation, the air mission table was removed without affecting the database integrity. Figure 12 depicts the tables derived from the entities shown in Figure 9.

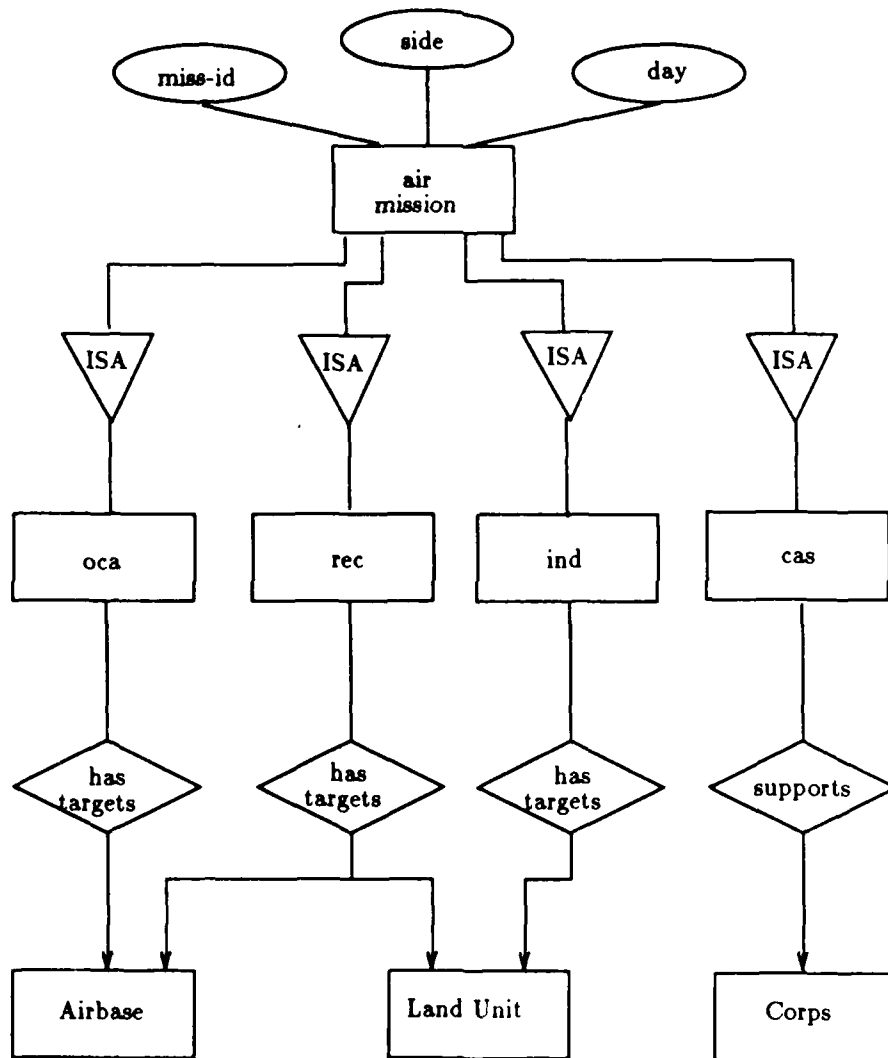


Figure 11. Sample E-R Diagram for "IS-A" Construct

Airbase Table

ab-id	ab-side	ab-name	...
50	blue	Ramstein	
63	blue	Bracknell	
75	red	SAFP AD	
52	red	EGAF AD	

Aircraft Table

ac-name	ac-side	sortie-rate	...
F4G	blue	5	
F111E	blue	3	
B52G	blue	2	
MIG17	red	6	

Aircraft-on-Airbase Table

ab-id	ab-side	ac-name	quantity
50	blue	F4G	15
50	blue	B52G	5
63	blue	F111E	14
75	red	MIG17	30
52	red	MIG17	20

Figure 12. Sample Tables Constructed From the TWX E-R Diagram

3.1.3 Data Dependency and Normalization Process. The next step in the logical design process was to determine the non-trivial functional and multivalued dependencies which held on the attributes for each candidate relation. As previously stated, if the E-R diagram was correctly designed, there should not be any non trivial MVDs which held on the attributes of any candidate relation. Also, for the TWX database, the primary key of each relation should functionally determine all other attributes within the relation. This proved to be true for each relation in the database, thus, the E- R diagram to candidate relation process worked very well for this project.

After identifying the data dependencies, the normalization process was started. The normalization process involved analyzing the various dependencies between the entity attributes and determining if the entity (relation) was in normal form. For example, a relation is in fourth normal form if for every non-trivial MVD ($X \twoheadrightarrow Y$), X is a superkey of the relation [5]. A superkey is a set of one or more attributes which allows you to uniquely identify an entity within an entity set [7]. However, superkeys are different from primary keys in that a primary key can be a subset of a superkey and are chosen by the database designer as the primary means of identifying entities in an entity set [7]. An example of a superkey would be the attributes "side, ab-id, ab-name" for the airbase relation as depicted in Figure 13. However, the primary key selected for this table was "side, ab-id" since it is the smallest subset of the superkey that still uniquely identifies the airbase records among the airbase entity set.

For the Micro-TWX database, the relations were normalized using fourth normal form (4NF) procedures. As pointed out by Korth and Silberschatz [7], it is advantageous to find a database design which is both 4NF and dependency preserving. A dependency preserving decomposition ensures there are no non-trivial FDs or MVDs which do not hold on some decomposed relation or are not implied by those FDs and MVDs that do hold. Dependency preservation allows testing of the dependencies by examining each relation in the database without doing expensive joins. Thus,

Airbase Table

ab-id	ab-side	ab-name	...
50	blue	Ramstein	
63	blue	Bracknell	
75	red	SAFP AD	
52	red	EGAF AD	

Superkey - ab-id, ab-side, ab-name

Primary key - ab-id, ab-side

Figure 13. Partial Airbase Table Denoting Relation Keys

the combination of 4NF and dependency preservation minimized the amount of data redundancy in the database while providing a means to efficiently maintain data integrity.

To complete the design process, it was necessary to ensure that all dependencies within the closure of the identified functional and multivalued dependencies for each relation were preserved, i.e., there were no non-trivial dependencies which did not hold on some relation. This proved true for all relations except the "orders" relation. The "orders" relation contains the HHQ tasking (move to new location, attack, etc) for the various land units within the land battle simulation. This table was not decomposed into a normal form because of some unique characteristics in the air and land simulation software. In order to properly decompose this table into a fourth normal form relation, significant modifications would have been required in the land simulation programs in order to process land order records. The benefit of putting this one table in 4NF did not warrant these changes.

3.2 Micro-TWX Database Implementation

The transformation of the logical database design into the physical TWX database proved to be a very straightforward process with the help of the various implementation tools available in the Ingres DBMS package. By using the Ingres menu driven development environment, each table

Airbase Table

side	ad-id	ab-name	num-shelters	...
blue	50	Ramstein	23	
blue	63	Bracknell	56	
red	75	SAFP AD	46	
red	52	EGAF AD	35	

Blue-Airbase Table

ab-id	ab-name	num-shelters	...
50	Ramstein	23	
63	Bracknell	56	

Red-Airbase Table

ab-id	ab-name	num-shelters	...
75	SAFP AD	46	
52	EGAF AD	35	

Figure 14. Example of the Horizontal Partitioning of the Airbase Table

was defined and created in the TWX master database. When completed, the new TWX database contained 123 separate tables and over 500 unique data attributes.

To help organize the database and to speed up query processing all tables which contained the "side" attribute (e.g., airbase, aircraft, etc) were horizontally partitioned by side. This has the effect of decreasing the amount of tuples in each of these relations by creating a separate table for each side (i.e., blue-airbase and red-airbase). Figure 14 depicts a horizontal partitioning of the airbase relation. A further horizontal partitioning of the tables by ATAF (2 and 4) was tried but the number of relations in the database (over 200) became unmanageable. Also, the benefits derived from having fewer tuples in each table was not enough to significantly reduce query processing time.

structure-type	query-type	number of recs
heap	sequential	small
heapsort	sequential	small
hash	exact match	large
isam	range	large
b-tree	range	large

Figure 15. Table of Ingres Indexing structures

After the relations were created and the initial exercise data was entered, an analysis of the types of queries and database updates required for each table in the TWX database was conducted. This was necessary to determine what type of index structures to create for each table in order to optimize query processing time while still minimizing the overhead for database updates.

The Ingres database management system provided several different types of indexing structures which were effective for various types of queries. Figure 15 shows the indexing structures available with Ingres and the type of database query for which they are most effective [1]. Each table created in the Ingres system automatically defaults to a heap structure. Records within a heap structure are random and unordered which requires Ingres to scan the entire table for each database query. This type of structure is not suitable for tables with a large number of records and requiring exact match or range queries on data in key columns.

The hash structure stores records randomly within a table based on some address determined by the value of a selected key [7]. This type of structure can significantly decrease query processing time for queries run on tables with a large number of records and requiring an exact match on some data key. For example, the `airbase` table is hashed on the key "ab-id" since most queries for this table are of the type "select * from airbase where ab-id = xxx"

Finally, the isam (indexed sequential access method) structure stores records in ascending order based on some key value and subdivides the records into pages. Then the largest key value

for each page is stored in a tree-structured index. This type of indexing is best used for range type queries, i.e., "select * from ac-on-airbase where quantity < xxx". However, there are some disadvantages associated with placing index structures on tables. First, more disk space is required to hold the indexes for each table. Each index will have as many rows as the table for which it is built [1]. Secondly, there is more processing overhead associated with each database update due to the need to create and modify indexes. These drawbacks had to be weighed against the query processing needs for each table. For the Micro-TWX database, most tables were indexed by the primary key using a hash structure. Then to support some of the user-interface queries, a secondary isam key was placed on attributes used in range type queries. However, the various air mission tables were left in a heap structure because the number of records in them would be small and a quick update time was desired to speedup mission input.

3.2.1 Micro-TWX System Layout. As mentioned, the Micro-TWX system must be able to support multiple seminars simultaneously. To accomplish this, the options were to store the data for each seminar in one large centralized database or create a separate database for each seminar. The advantage of using a centralized database scheme would be that any global data updates necessary for all seminars could be made at one time versus the controller having to make a separate update for each individual seminar. Also, the amount of disk space required for the entire TWX system would be less since the internal Ingres data structures would not have to be repeated for each seminar.

However, there are disadvantages associated with this method. First, query processing time would be slower because the increased number of records in each table. Also, the "seminar-number" attribute would have to be added to every table in order to uniquely identify each seminar's records. Finally, when using a centralized scheme, it would be difficult to restore a particular seminar's records without restoring the entire database, thus impacting all the seminars playing the exercise.

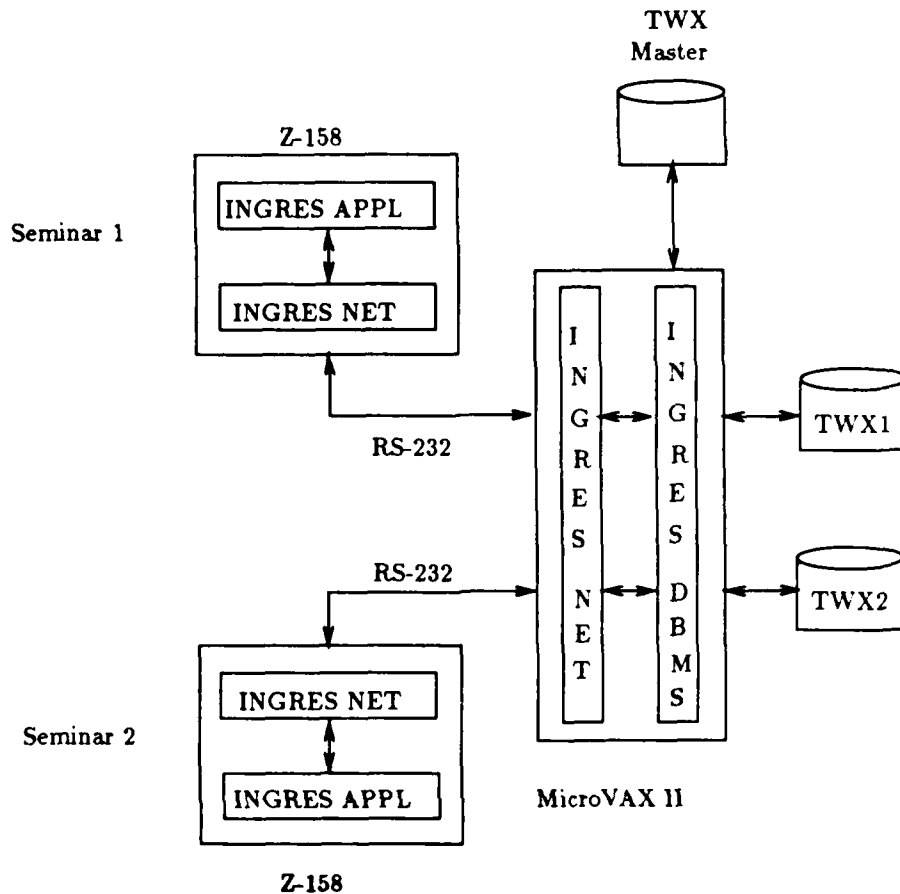


Figure 16. Micro-TWX System Layout

Because of the problems associated with a centralized scheme a separate database will be created for each seminar based on a generic TWX master database.

Prior to day 1 of the exercise, the game controller will make a copy of the TWX master for each seminar. Then, if desired, the individual seminar databases can be tailored by using the Ingres Query-By-Form menu driven update capability to add airbases, land units, munitions, etc. Also, a master database could be created for any particular theater, (e.g., Korea, mid-east, etc) which adds a great amount of flexibility and creativity to the exercise. Figure 16 depicts the Micro-TWX system layout.

IV. Micro-TWX Air Battle Simulation Software Conversion

This chapter addresses the issues concerned with transferring the current TWX air simulation software from a mainframe to a microcomputer environment. The goal of this effort was to incorporate a new mission targeting scheme and database system into the air simulation using a upgraded version of the Fortran compiler (Fortran 77) and the Ingres DBMS host language interface. These changes had to be accomplished with minimal impact on the basic air battle algorithm which proved to be a difficult but not impossible task. Secondly, the Micro-TWX air simulation software validation is presented showing that the output results are consistent with the current TWX system. The combined integrated system validation is addressed in a separate thesis writeup [8]. Also, because of its size and complexity, the software conversion process was the most difficult part of this thesis project, requiring over 8 dedicated weeks to complete.

4.1 Air Battle Simulation Software Overview

As previously stated, the current TWX air battle simulation is a Fortran IV program which runs on a 1970's vintage Honeywell mainframe computer. The simulation itself is a large scale, aggregated air model that simulates the interactions between opposing offensive and defensive air forces during a five day theater-level conflict [2]. The outcomes of the air battles are determined separately for each ATAF (2 and 4) and for each cycle (day and night). There are nine different mission types which are simulated in the model and they can interact in various ways. For example, flying defense suppression missions can reduce the number of ground-to-air losses at the FEBA (forward edge of battle) for other mission types, i.e., offensive counter air and interdiction. Figure 17 depicts the various mission types and their function.

The original TWX air simulation software consisted of approximately 150 separate programs¹ containing over 9000 lines of code. This did not include the land simulation and user interface

¹Several functions within the current air simulation are performed by Honeywell system routines and had to be created for the Micro-TWX version

Mission Type	Function
Offensive Counter Air (OCA)	Attack enemy airbases
Interdiction (IND)	Attack enemy rear echelon
Battlefield Air Interdiction (BAI)	Attack enemy land units
Close Air Support (CAS)	Support friendly ground units at the FEBA
Defense Suppression (DEF)	Destroy enemy ground-to-air defenses
Defensive Counter Air (DCA)	Intercept enemy OCA air missions
Electronic Counter Measures (ECM)	Provide ECM support
Combat Air Patrol (CAP)	Intercept attacking enemy aircraft at the FEBA
Reconnaissance (REC)	Provide recce of enemy airbases and land units

Figure 17. Mission Types Modeled by the Air Simulation

Name	Function
CI - Cycle Initialization	Reset cycle dependent arrays
CU - Cycle Update	Update database after completion of each cycle
DI - Day Initialization	Initialize Day dependent common arrays
DO - Day Output	Create end of day simulation reports
DU - Day Update	Update the database after completion of both cycles
FI - Fly Initialization	Allocate sorties for requested air missions
FL - Fly	Run the air model using the requested red and blue missions
FO - Fly Output	Print air battle results for blue and red
FU - Fly Update	Reset FLY arrays for next ATAF
PI - Program Initialization	Initialize the simulation common arrays
UT - Utility Library	Contains TWX utility routines
WU - Wrap Up	Cleans up database and sets up land interface

Figure 18. Air Simulation Functional Areas

software which had to be separated from the air simulation code. The air simulation routines are divided into 12 unique functional areas that are functionally or temporally bound [2]. Figure 18 outlines each functional area and its purpose in the simulation. This functional breakout maps directly into the main air algorithm as shown in the flowchart depicted in figure 19.

Overall, the air simulation documentation for the current system was very comprehensive and an invaluable asset for accomplishing this project [2]. However, the software code itself contained very few programmer comments and was mostly written in reverse logic so that a pseudo if-then-else structure could be implemented (Fortran IV did not support if-then-else or do while statements).

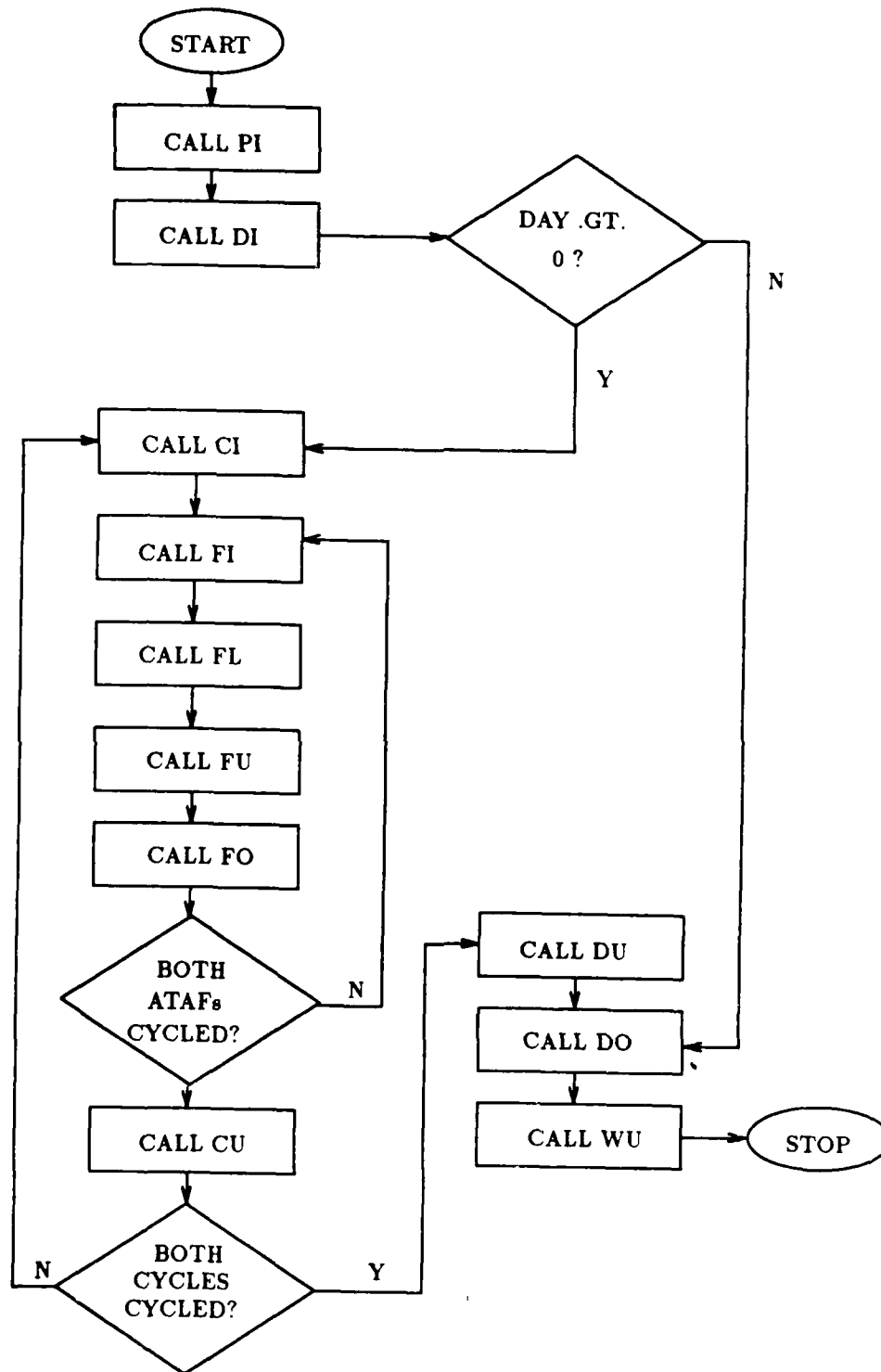


Figure 19. Air Simulation General Flowchart

Although this was a good programming style when using the old Fortran IV compiler, there were some aspects which are not compatible with the new Fortran 77 compiler used for this project. As a result, most subroutines were rewritten to remove this reverse logic in order to make the code easier to read and understand.

Another aspect of the current TWX software was that it used common blocks of array structures to hold the exercise data during the simulation. Common blocks are Fortran global data structures which makes the inter-routine communication easy and allows fast data access since the data is memory resident at all times. However, using this type of global data structure caused some significant debugging problems since the global data can be accessed by any routine. This was particularly true in this conversion since the "fly" common blocks and subroutines were extensively changed and it was often difficult to pinpoint the problem routine when errors arose.

4.2 Simulation Conversion Process

The conversion process started by separating the subroutines unique to the air simulation program from the user interface and land battle simulation. This required a thorough analysis of the entire TWX software because many of the routines were common to both the user interface and air simulation. Next, an extensive evaluation of the common block array structures was conducted to analyze the changes necessary to increase the number of airbases, aircraft types, and munition types per side. This enhancement was accomplished by increasing the array bounds for each common block and changing the simulation parameters where necessary.

The next step was to pick a place to start with the actual software conversion process. Since the simulation software was already divided into functional areas, the logical place to begin was the program initialization (PI) routines and proceed through each functional area as it was called in the flowchart in Figure 19. This procedure worked very well since each functional area could be debugged and tested prior to starting on the next functional area. Also, another advantage to

using this method was that the simulation data was set up properly in the common data blocks in order to test each successive functional area. To support the testing of each area, the Ingres DBMS Query-By-Forms data entry system was used to dynamically input test case air missions and simulation data. Without this capability, the fly routines could not have been tested until the air simulation was fully integrated with the user interface software.

In order to incorporate the new database system into the air simulation software, every application file reference within each subroutine had to be replaced with embedded Ingres SQL (Structured Query Language) statements. Embedded SQL is the Ingres DBMS unique statements which are placed inside of the Fortran subroutines to access the database. Prior to compiling the Fortran source code the embedded statements had to be processed through an Ingres DBMS precompiler to be converted into actual Fortran 77 statements. An example of the conversion process is shown in Figure 20. Here, the old statements 6-10 in part A are replaced by the Ingres embedded SQL statements 1-19 in part B. Although the old application file system using Honeywell random access methods to retrieve and store data was faster (no query interpretation required), the Ingres DBMS method of using embedded SQL statements is easier to understand and is not physically or hardware limited like the old version.

One particular problem encountered, during this conversion process was that in the current air simulation software the 384 simulation constants were hardcoded into a common array buffer. These constants are referenced using specific indexes in numerous subroutines throughout the simulation. In order to preclude changing each of these routines, the air constant relations within the new database were modified to provide a mapping index into the common buffer array. Figure 21 shows the air-constants table as derived by the logical design process outlined in Chapter 3 and the new table layout required to support the current system.

Another problem which arose in the conversion concerned how the aircraft types and munition types are mapped into their respective common arrays. In the current TWX system the airbase,

PART A

```

001  READ(11'1)IRECSIZE,IPOINT(2,1),NRECS(2,1),IPOINT(2,2),NRECS(2,2)
002  DO 300 IS=1,2
003      IORIGIN=IPOINT(2,IS)-1
004      DO 200 IA=1,NRECS(2,IS)
005          IREC=IORIGIN+IA
006          READ(11'IREC)ACNAME(IA,IS),BUF
007          SRATE(IA,IS)=BUF(1)
008          SFACT(IA,IS)=BUF(2)
009          STK(IA,IS)=.FALSE.
010          IF(INT(BUF(25)+0.5).NE.0) STK(IA,IS)=.TRUE.
011 C      ENDDO
012 300  CONTINUE
013 C  ENDDO

```

PART B

```

001 EXEC SQL DECLARE BLCSR CURSOR FOR          ! Select blue AC
002 1 select ac_name,ac_role,surge_fac,
003 1 sortie_rate
004 1 from aircraft

005  EXEC SQL WHENEVER SQLERROR CALL ING_ERROR

006 EXEC SQL WHENEVER NOT FOUND GOTO 400      ! EXIT LOOP WHEN OUT OF DATA

007  DO WHILE ( SQLCODE .EQ. 0 )
008      EXEC SQL FETCH BLCSR INTO
009 1  :AC_NAME,:AC_ROLE,:SURGE_FAC,:SORTIE_RATE,
010 1  :SPARES_SORTIE

011      AC_TYPE = AC_NAME//AC_ROLE           ! concat name and role
012      CALL UTAC_INDEX(AC_TYPE,SIDE,INDEX)   ! find index in mapping array
013      SRATE(INDEX,SIDE) = SORTIE_RATE       ! set sortie rate
014      SFACT(INDEX,SIDE) = SURGE_FAC        ! set surge factor
015      IF (AC_ROLE .EQ. 'S') THEN           ! strike (nuc) a/c ?
016          STK(INDEX,SIDE) = .TRUE.
017      ELSE
018          STK(INDEX,SIDE) = .FALSE.
019      END IF
020  END DO

```

Figure 20. Example Showing Ingres Embedded DBMS Calls

Original Table Format

frac-ac-grnd	max-ecm-con	min-ecm-con	...
.321	.4	.52	

New Table

constant-name	array-index	value
frac-ac-grnd	1	.321
max-ecm-con	2	.222
min-ecm-con	3	.55
...

Figure 21. TWX Air Constants Table

land unit, and munition names are hardcoded inside the air simulation itself on a one-to-one correspondence with the application data files. This required the TWX air software to be modified each time an airbase, land unit, or munition type was added or deleted and the entire simulation had to be recompiled. In order to provide database independence from the position dependent simulation arrays, a mapping subroutine was written for both the aircraft and munition entities since these entities are often referenced by name. This mapping provides a one-to-one correspondence between the array index and the aircraft or munition name using a binary search algorithm.

4.2.1 Converting the Mission Targeting Scheme. The one air simulation enhancement requested by the Air Wargaming Center, besides updating the current file system, was to change the way air mission packages are formatted during ATAF mission input. Under the current system, air missions are input by aircraft type with multiple targets and associated sorties assigned by line. Figure 22 shows the current offensive counter air mission input format. However, the Wargaming Center wanted to incorporate a more realistic mission packaging concept where each line represented a target with multiple aircraft types and associated sorties. This approach reflected a more realistic "real world" situation where the target is the main focus in air mission planning and not the aircraft type. Figure 23 depicts the new ATAF mission input format implemented by this

OCA - Offensive Counter Air Missions

Mission A/C							Escort		Def Sup		ECM	
Miss Num	A/C Type	Tar Num	No Sort	Tar Num	No Sort	...	A/C Type	No Sort	A/C Type	No Sort	A/C Type	No Sort
1000	F4 A	38	10	40	15		F15 D	4	F4 G	2	F111 E	1
1001	F16 A	38	10	32	10		F4 D	4	F4 G	2		
1002	A7 A	20	12									
1003												

Figure 22. Current TWX Mission Input Format

OCA - Offensive Counter Air Missions

Mission A/C							Escort		Def Sup		ECM	
Miss Num	Tar Num	A/C Type	No Sort	A/C Type	No Sort	...	A/C Type	No Sort	A/C Type	No Sort	A/C Type	No Sort
1000	38	F4 A	10	F16 A	10		F15 D	4	F4 G	2	F111 E	1
1001	40	F4 A	15				F15 D	2	F4 G	1		
1002	32	F16 A	10				F4 D	2	F4 G	1		
1003	20	A7 A	12									
1004												

Figure 23. New TWX Mission Input Format

project.

Incorporating this enhancement into the air simulation software created several problems in the conversion process which were difficult to overcome. First, the original consensus was that the conversion process could be accomplished without the need to understand the complex air simulation algorithms. This proved to be wrong, because to change the targeting software within the air simulation, a thorough understanding of the fly algorithms was required in order to modify the array structures and allocate the escort aircraft for each mission. This was critical since one of the main goals of the conversion was to ensure output results consistent with the current TWX simulation.

Secondly, there were two options available to remap the mission targeting to the new format. One option was to forge the new mission input data into the old mission input format and then use the existing fly routines to process the missions. This seemed to be an attractive solution, however,

the process became overly complicated when trying to remap the final simulation results back into the new mission format. Although this could have been accomplished, future maintenance and enhancements would be difficult to implement due to the complexity and constraints imposed by this method. As an alternative, the selected approach was to completely remap the existing air simulation common block array structures to conform with the new mission input. However, this required each "FI", "FL", and "FU" subroutine to be completely rewritten. Even though this method added approximately 3 weeks to final completion date of the air simulation, it will be much easier to understand and maintain in the future.

4.3 Air Battle Simulation Validation Process

Since there was never a formal validation conducted against the original TWX air simulation algorithms, the only way to validate the new Micro-TWX version was to use the current version as a baseline. According to Roth [11], the original version was "tuned" until it produced the desired results for any particular air battle. Consequently, the same process must be used for the new Micro-TWX air simulation, since a formal validation would be a thesis effort unto itself.

As stated in Chapter I, the air simulation would be considered valid if it produced results consistent with the current air simulation. Since the air simulation was extensively modified to incorporate the the new mission targeting format, exactly matching results were not expected. Also, since the computer architectures are significantly different, the air simulation random number generator could generate slightly different values even though the same seed is used. This could cause some significant differences in the final results if the actual weather (i.e., good, bad, or poor) for the two simulations are different from each other. Actual game weather for a particular weather zone is computed by a probabilistic algorithm.

In order to properly compare the two air simulations, each TWX system had to be set up with the same initial data and air constants. Also, a mapping was required between the two different

mission input formats so that the air missions are as equitable as possible. For example, the escort aircraft must be allocated such that they have the same effect in each simulation. Given these assumptions, tests were run on both systems.

Based on preliminary runs, the new Micro-TWX system produces results consistent with the current version. However, a careful analysis needs to be accomplished where the effects of varying the number of ECM, Defense Suppression, Defensive Counter Air, and Combat Air Patrol missions remains constant between the two simulations. This extensive testing could not be accomplished within this thesis effort due to time constraints caused by the additional time required to incorporate the new mission packaging changes.

V. Micro-TWX Controller Application

This chapter discusses the development of a prototype exercise controller application for the Micro-TWX system. The purpose of this application was to provide a way for the TWX game controller to create, maintain, and monitor the various seminar exercises from a centralized location. By using a VT200 series terminal connected to the MicroVAX computer, the controller can oversee the various seminar exercises, ensuring each seminar runs as smoothly as possible. The following sections discuss the requirements and design issues associated with developing this prototype application.

5.1 Functional Requirements

The purpose of the controller function within the Micro-TWX system is to provide a centralized environment from which an exercise controller can initiate, monitor, and maintain multiple seminar exercises. In order to accomplish this task, the environment must be a user friendly, menu driven application which does not require the controller to master the host computer's operating system commands. The following is a list of the capabilities required for this application as stated by the Air Force Wargaming Center and how each one was implemented in this application. In order to provide a reference, Figure 24 shows the main menu and associated functions available within the prototype controller application.

- 1. Provide the ability to monitor and control each seminar.**

- (a) Must be able to control player access to the AAFCE and ATAF menus.**

This was accomplished by using AAFCE and ATAF access flags in each seminar database.

The user interface application checks the appropriate access flag prior to allowing entry by the player. The exercise controller can use the database update option to set these flags as required.

TWX CONTROLLER FUNCTION

=====

1. Create a new seminar database
2. Destroy an existing seminar database
3. Review or Update an existing TWX database
4. Print Reports
5. Start a simulation
6. Review or update database access
7. Backup or Restore a database

+-----+
|Please enter selection: |
+-----+

QUIT(1) HELP(2) Check Queue(3) Monitor Seminar(4) Monitor Sim(5)

Figure 24. Micro-TWX Controller Application Main Menu

- (b) **Must be able to terminate seminar input or play without prior notice to the exercise players.**

This is accomplished by using the same access flags described in the previous item. However, if a user is already inside a function (AAFCE or ATAF) they will not be immediately terminated. Only when they exit will the flag take affect. This was necessary to preclude any input data from being corrupted and creating an inconsistent database.

- (c) **Provide the capability for two-way message traffic between the game controller and each seminar currently logged on.**

This item has not yet been implemented. However, depending on the final TWX system configuration there are two ways to implement this capability. First, if the user interface is running on the Z-158 using the Ingres-NET software, there is no way to use the MicroVAX operating system utilities for communication. Therefore, special tables would have to be created in both the seminar database and controller database which would contain message traffic. Also, special flags would have to be created in the database to alert the players and controller of message transactions.

However, if the user-interface software is run on the MicroVAX with the Z-158's emulating a VT200 terminal, the MicroVAX operating system utilities can be used for communication. Here, the "phone" utility could be used for concurrent, two-way dialog between the player and controller or the "mail" utility could be used to send messages back and forth between players and/or controller.

- (d) **Provide indicators to the controller when a seminar completes a major function (i.e., AAFCE, ATAF, etc)**

This is accomplished by the user interface software setting specific flags within the seminar database when the players are finished with each major function. The controller can monitor these flags using the "Review or update an existing TWX database" function.

- (e) **Provide the capability to monitor the progress within each seminar. For example, see re-role option currently activated in AAFCE for seminar 1 or see seminar 2 currently inputting REC missions in ATAF.**

This capability is provided by the "Monitor Seminar" function. This option drives a form which displays the current status of each seminar.

- (f) **Provide an indicator which indicates that a seminar has finished all input (both AAFCE and ATAF) and is ready to run the simulation.**

Again, this is accomplished by the user interface application setting the appropriate database flags and the controller monitoring those flags.

2. Provide the ability to start and monitor each simulation.

This is accomplished by the "Start a simulation" function.

- (a) **Provide the ability to tell if the simulation executed properly. If there are problems, provide an explanation.**

This can be accomplished by checking the final output listing produced by the operating system. The simulation has numerous informational error messages built in regarding database errors.

- (b) **Provide a way to control and monitor each individual seminar's printout (simulation reports).**

This capability is provided by the "Print reports" option.

3. Provide a means to make on-the-fly database updates for any seminar.

This capability is provided by the "Review or update an existing TWX database" option.

4. Provide a restart capability for any individual seminar.

This can be accomplished by the controller making a backup each day of the individual seminar databases using the "Backup" function. If any particular seminar needs to be restarted,

the "Restore" function can be used to put the desired seminar database back to known state for any particular day.

5.2 Design Approach

In order to meet the stated requirements, the controller application must perform a variety of functions which interact with both the Ingres database management system and the MicroVAX operating system. To meet this demand, the application was developed using the Application-By-Forms feature of the Ingres database management system. This feature is an application development tool which integrates a forms editor, fourth generation language (4GL), and source code manager into a total application development environment. By using this programming development tool, a functional, prototype controller application was developed in minimal time. Also, since it is very easy to make changes to the forms and/or associated 4GL programs, any required functions not implemented by this thesis project can be easily added at later time.

5.2.1 Design Overview. The controller application will be automatically invoked when the controller signs onto the MicroVAX system using the correct username and password. This was accomplished by running the application from the controllers login command file which is automatically executed by the operating system each time a user signs onto the system.

The prototype controller application is composed of main menu and 2 submenus. In order to minimize the number of menus (forms) required to accomplish each function, the main menu uses the bottom section of the screen to prompt for required input and display informational messages. This allows the user (game controller) to always be aware of what function he/she has selected and what other functions are available. From the main menu, a function can be selected by entering the appropriate number or by pressing a function key. Also, a help option (function key 2) is available which provides a tutorial on the controller function. To terminate the application, the user presses "Quit" (function key 1). This closes down the controller function and automatically

logs the controller off the system.

The Micro-TWX Controller Application maintenance manual contains the associated menus, 4GL code, and predefined command files. A command file is a MicroVAX text file which contains MicroVAX operating system commands. Since the MicroVAX operating system is a multiprocessing system, these files can be submitted to the operating system for concurrent execution in a background mode. The following list describe each function in the main menu and how it was implemented.

1. Create a new seminar database.

This function will create and initialize a new seminar database using the TWX master database as input. The following sample dialog is initiated at the bottom of the form when the user selects this option.

Please input the seminar number to be created: 5

A database called TWX5 will be created; Proceed (Y/N)?:

If the user enters "N", the message "***** Action Aborted *****" will be displayed and the menu reinitialized for a new selection. This allows the user to verify the seminar number prior to creating the database.

If the answer is "Y" a system command is executed which submits a predefined command file as a background (batch) process to the MicroVAX operating system. This process will run independent of the controller application and will create and initialize the seminar database using Ingres system commands. The user can monitor the progress of this process by using the "Check Queue" function key. This function will display the processes currently running in a batch mode on the MicroVAX system. When the process is completed, the controller can verify the results by checking the process output listing generated by the operating system.

2. Destroy an existing seminar database

This function will destroy an existing seminar database by using a 4GL system call to invoke the Ingres "DESTROYDB" command. It's dialog is the same as in selection 1. This will allow the controller to clean up the MicroVAX system after the exercise is concluded. Also, this is an interactive function and does not submit a batch process.

3. Review or update an existing TWX database

This function will prompt for the desired seminar database name and will call the Ingres Query-By-Forms (QBF) application. This application is a user friendly environment which will allow the exercise controller to make dynamic database updates or deletions via predefined forms. For example, in order for a controller to control a seminar's entry into the AAFCE and ATAF function, special seminar database lockout flags must be set using this option.

4. Print Reports

This option will prompt for the desired seminar number and print the air and land simulation reports for that seminar. The "Check Queue" option will allow the controller to monitor the various print jobs in the system.

5. Start a simulation

This function will start the air and land simulation for a particular seminar. The following dialog is initiated at the bottom of the form when the user selects this option.

Please input the seminar number to be started: 5

The simulation for seminar 5 will be started; Proceed (Y/N)?:

If the user enters "N", the message "***** Action Aborted *****" will be displayed and the menu reinitialized for a new selection. This allows the user to verify the seminar number prior to starting the simulation.

If the answer is "Y" a system command is executed which submits a predefined command file as a background (batch) process to the MicroVAX operating system. This process will

run independent of the controller application and will start the air and land simulations for the selected seminar. The controller can monitor the progress of the simulation by using the "Check Queue" function key. This function will display the processes currently running in a batch mode on the MicroVAX system. When the process is completed, the user can verify both the air and land simulation results by checking the output listing generated by the operating system. Also, the Monitor Simulation (function key 5) allows the controller to preview a selected simulation's log file to check it's current status. For example, within the air simulation each time a major function is started, i.e., "PI", "DI", etc, a print statement is written to the log file stating the current function and start time. This way, the controller can monitor the progress of a simulation and tell approximately when it is going to finish.

6. Review or update database access

This is a database administrator function which invokes the Ingres "accessdb" command via a 4GL system call. This command is a forms based interface to list and modify the various databases under controller supervision. The controller would use this option to get a list of active databases.

7. Backup or Restore a database

This option provides the capability to backup a seminar database to tape or restore a database from tape. The submenu in Figure 25 is called first to prompt for the desired option. The following dialog is initiated at the bottom of the screen when the user selects the "Backup" option.

Please enter the seminar number to backup: 5

Please enter the label name for the tape: TWX5DAY1

A backup of TWX5 using label name TWX5DAY1 will be made. Proceed (Y/N):

TWX DATABASE BACKUP AND RESTORE FUNCTION

1. Backup a TWX database
2. Restore a TWX database

```
+-----+  
|Enter Selection: |  
+-----+
```

Figure 25. Backup and Restore Menu

If "N" the message "***** Action Aborted *****" will be displayed and the menu reinitialized for new input. This allows the controller to verify the seminar number and cancel if necessary.

If the answer is "Y", the application will submit a predefined command file to the MicroVAX operating system which will backup the selected seminar database to tape. However, the controller must know how to load and label the tape. The controller may monitor the batch job by using the "Check Queue" option (function key 2). Function key 1 will quit this menu and return the user to the main menu.

The "Restore" option dialog is similar to the "Backup" option. However, this option will restore a seminar database from tape. Since this is a full restore, any existing data in the database will be overwritten.

8. Monitor Seminar (Function key 4)

This option will display a table showing the current function of each seminar. Figure 26 shows the table with sample data. Function key 2 "Refresh Table" will update the table values while function key 1 "Quit" will return the user back to the main menu.

TWX MONITOR FUNCTION

Sem Num	Side	Current Function	Total Time
01	B	AAFCE LOG AIR	20 mins 23 secs
01	R	2ATAF OCA MSN	10 mins 43 secs
02	B	4ATAF BAI MSN	20 mins 12 secs

QUIT (1) REFRESH TABLE (2)

Figure 26. Monitor Seminar Table with Sample Data

As shown, the prototype application is completely menu driven and does not require the controller to master the MicroVAX operating system commands. However, there are several functions outside of the TWX system which need to be performed by a "system manager" which could not be included in this application. For example, the system manager would be responsible for creating the various user accounts necessary to conduct the exercise and also with monitoring the overall status of the MicroVAX system. This individual should be a software specialist familiar with the MicroVAX operating system and architecture.

VI. Conclusion and Recommendations

This chapter provides a summary of the issues concerned with transporting a large scale computer wargame exercise from a mainframe to a microcomputer environment. This includes integrating a new database system to support a more friendly user interface and converting the air simulation software. Also, recommendations for additional work and enhancements are discussed pointing out various areas which need attention.

6.1 Summary

The Theater War Exercise is a large scale, wargame simulation conducted by the Air War College to train senior level Air Force officers in the application of airpower. Even though the TWX computer software was developed almost a decade ago, the exercise continues to be an integral part of the Combined Air Warfare course and other Air War College curriculums. However, with the advances in computer hardware and software technology, TWX is rapidly becoming obsolete due to inherent problems associated with using old computer equipment and an unfriendly user interface. The goal of this thesis effort was to bring the TWX system up to modern standards in both software and hardware capabilities while providing increased exercise portability. System portability was desired so the exercise could be played at remote sites without the need for mainframe computers and Air Force Wargaming Center personnel to conduct each exercise.

In order to provide system portability, the Micro-TWX system was targeted to a combination of IBM PC/XT compatible microcomputers and a Digital Equipment Corporation MicroVAX II. By using the Ingres network software, the IBM/PC computers can run the user interface software while the MicroVAX acts as a fast database fileserver and simulation host. This setup is powerful enough to support up to 8 simultaneous users such that up to 4 separate exercises could be running concurrently.

The most important goal of this thesis was to improve the TWX user interface. Here, the Ingres relational database management system was used to replace the application dependent file system currently used in the Honeywell TWX system. The Ingres DBMS provided the application development tools necessary to develop a more user friendly interface, incorporating help facilities, on-line editing, and real-time data validation. Also, by using the DBMS to perform the TWX data manipulation, a tremendous amount of flexibility for maintaining and enhancing the TWX system was obtained.

In order to implement the new Micro-TWX database, a relational design approach was used which combined the top-down method using conceptual design tools (semantic models) and the bottom-up method using traditional data dependency and normalization techniques. This logical design approach had numerous advantages over other relational design methods and was used with outstanding success.

Once the new database was designed and implemented the air simulation software was converted. This involved removing the current application dependent I/O statements and substituting the Ingres DBMS embedded database calls. Also, the mission targeting format was extensively modified to reflect a new "mission packaging" concept. After the simulation was converted and running, it was tested using the old Honeywell TWX system as a baseline. Preliminary simulation results proved consistent between both systems.

Finally, a Micro-TWX controller application was developed using the Ingres Application-By-Forms application development tool. This application will provide a user friendly environment from which the game controller can initiate, monitor, and maintain multiple seminar exercises.

Overall, the new Micro-TWX system is a significant improvement over the old Honeywell system. By improving the user interface, the exercise players can now concentrate on playing the exercise rather than mastering the computer environment. Also, the game controllers now have a tremendous amount of flexibility in modifying the various seminar databases in order to

make the exercise more responsive. This, along with the fact any additional enhancements will be significantly easier to implement made this project well worth the effort and expense.

6.2 Recommendations for Further Work

In order to meet the deadlines required for this project certain aspects within the conversion process had to be placed aside. As stated, incorporating the new mission targeting format into the air simulation required additional time which was initially allocated to other sections of this conversion process. As such, the following items should be addressed by the Wargaming Center personnel who inherit Micro-TWX.

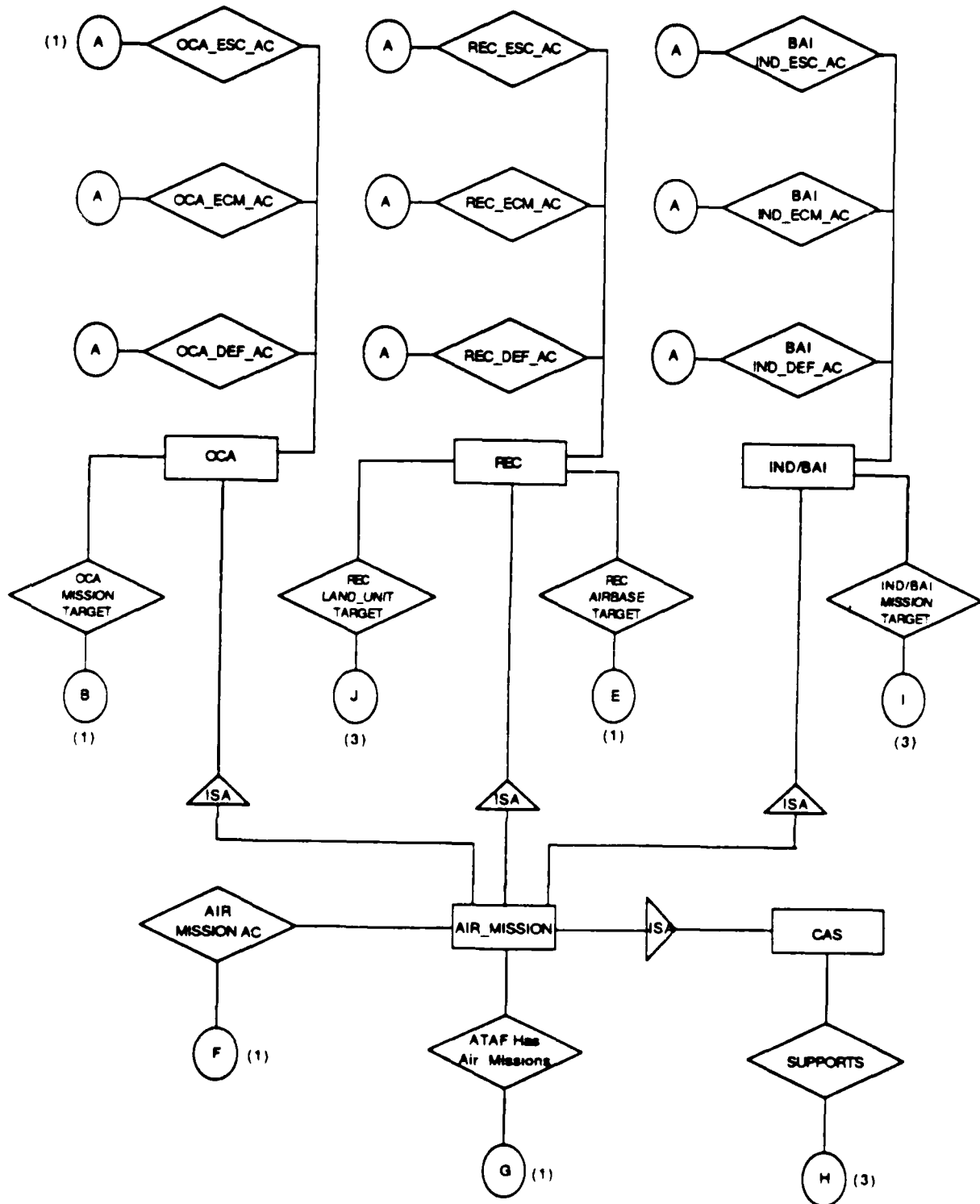
1. Incorporate the statistical routines into the air simulation. These routines were not necessary for the actual air model and were not converted. However, the necessary "hooks" have been incorporated and the implementation process should be straightforward.
2. Verify the initial exercise data within the TWX master database. Although great effort was used to ensure the "correctness" of the data, the data should be verified by TWX Wargaming personnel.
3. An extensive validation needs to be performed on the new Micro-TWX air simulation using the Honeywell TWX system as a baseline. This should be accomplished prior to using it in an actual exercise. For example, the defense suppression air mission type can have an affect on the number of ground-to-air losses incurred by offensive counter air, interdiction, battle field interdiction, and close air support missions at the FEBA. In order to verify the new model truly reflects the old model, the number of defense suppression sorties should be varied while the other air missions are held constant over a number of separate runs. Then the output results from both systems should be compared to verify they are consistent (i.e., consistent in both numbers and trends).

As an additional recommendation, a thorough analysis of the TWX air battle simulation should be conducted against existing air combat models. Although the exercise does not claim to reflect reality, enough emphasis is placed on the battle itself to warrant the effort necessary to verify it's combat validity. Finally, the TWX personnel should look at enhancing the system to incorporate real-time graphical displays. This capability would be especially beneficial for the land battle FEBA plot which is currently produced only on hardcopy printouts.

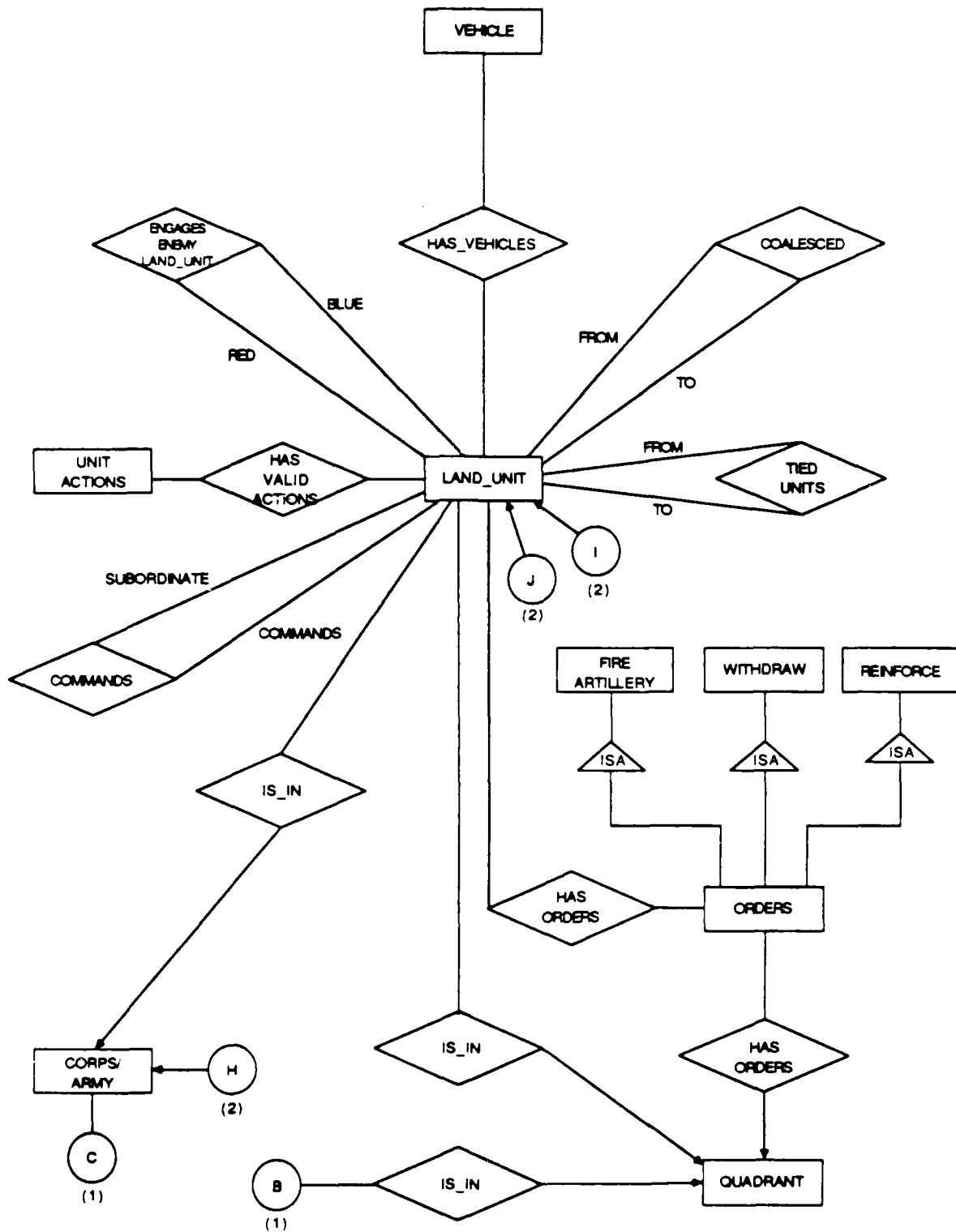
Airbase-Aircraft-PMS (Section 1)
 (*) indicates associated section number



Air Mission (Section 2)



Land -Units, Quadrants (Section 3)



Constants (Section 4)

Seminar
Control

Air Battle
Constants

Land Battle
Constants

Appendix B. Relations Derived from E-R Diagram

The following tables are the initial database relations derived from the E-R diagram in Appendix A. The final Micro-TWX database was horizontally partitioned by the side attribute for the airbase, aircraft, pms, air mission, and land unit relations. The key attributes for each relation are italicized.

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>ATAF</i>	I-1
<i>ABJD</i>	I-1
<i>AB_NAME</i>	C-20
<i>AB_STATUS</i>	F-4
<i>AB_X_CORRD</i>	F-4
<i>AB_Y_COORD</i>	F-4
<i>AB_CAPABILITY</i>	F-4
<i>INTELINDEX</i>	F-4
<i>MAX_TONNAGE</i>	F-4
<i>NUM_SHELTERS</i>	I-2
<i>NUM_REVETMENTS</i>	I-2
<i>RAMP_SPACE</i>	F-4
<i>TGT_HARDNESS</i>	I-2
<i>TGT_POP_DAY</i>	I-2
<i>OTHERINDEX</i>	I-2
<i>OVERRUN_STATUS</i>	I-1
<i>DEFINDEX</i>	F-4
<i>TGT_EFFECT</i>	F-4
<i>BS_LOAD_FAC</i>	F-4

Table 1. Airbase Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>AB_ID</i>	I-1
<i>EXERCISE_DAY</i>	I-1

Table 2. Valid Airbase Flydays

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AC_NAME</i>	C-4
<i>AC_ROLE</i>	C-1
<i>SURGE_FAC</i>	F-4
<i>ABORT_RATE</i>	F-4
<i>AIR_TO_AIR_CAP</i>	F-4
<i>DAY_SURV_FAC</i>	F-4
<i>NIGHT_SURV_FAC</i>	F-4
<i>RAMP_SPACE</i>	F-4
<i>SPARES_PER_SORTIE</i>	F-4
<i>POL_PER_SORTIE</i>	F-4
<i>SORTIE_RATE</i>	F-4
<i>ALL_WX_CAP</i>	F-4

Table 3. Aircraft Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AC_NAME</i>	C-4
<i>AC_ROLE</i>	C-1
<i>CAP_IN_TONS</i>	F-4

Table 4. Cargo Aircraft Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AC_NAME</i>	C-4
<i>AC_ROLE</i>	C-1
<i>SENSOR_TYPE</i>	I-1

Table 5. Recce Aircraft Sensor Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AB_ID</i>	I-1
<i>AC_NAME</i>	C-4
<i>AC_ROLE</i>	C-1
<i>QUANTITY</i>	I-2

Table 6. Aircraft on Airbase Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AB_ID</i>	I-1
<i>AC_NAME</i>	C-4
<i>AC_ROLE</i>	C-1

Table 7. Aircraft Allowed on Airbase Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AC_NAME</i>	C-4
<i>FROM_ROLE</i>	C-1
<i>TO_ROLE</i>	C-1

Table 8. Valid Aircraft Rerole Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AC_NAME</i>	C-4
<i>AC_ROLE</i>	C-1
<i>MISSION_TYPE</i>	I-1
<i>WX_TYPE</i>	I-1
<i>PREFERED_LOAD</i>	I-2

Table 9. Preferred Munition Load Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AC_NAME</i>	C-4
<i>AC_ROLE</i>	C-1
<i>MISSION_TYPE</i>	I-1
<i>CYCLE</i>	I-1
<i>WX_TYPE</i>	I-1
<i>DESTRUCTIVE_INDEX</i>	F-4

Table 10. Destructive Index Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>PMS_NAME</i>	C-6
<i>AIR_WEIGHT</i>	F-4
<i>SURFACE_WEIGHT</i>	F-4
<i>DISPLAY_ORDER</i>	I-1

Table 11. POL, Munitions, and Spares Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AB_ID</i>	I-1
<i>PMS_NAME</i>	C-6
<i>QUANTITY</i>	I-4

Table 12. PMS on Airbase Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AB_ID</i>	I-1
<i>PMS_NAME</i>	C-6
<i>PREDIRECT_RATE</i>	I-4

Table 13. Airbase PMS Predirect Rate Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>LOAD_NUMBER</i>	I-2
<i>MUNITION_NAME</i>	C-6
<i>QUANTITY</i>	I-2

Table 14. Standard Munition Load Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>CORPS</i>	I-1
<i>WX_ZONE</i>	I-1
<i>INTEL_INDEX</i>	F-4
<i>TGT_EFFECT</i>	F-4
<i>CMBT_PWR_RATIO</i>	F-4
<i>DEF_INDEX</i>	F-4

Table 15. Corps Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>UNIT_ID</i>	I-2
<i>UNIT_NAME</i>	C-20
<i>UNIT_TYPE</i>	I-1
<i>CURR_CMBT_PWR</i>	F-4
<i>CORPS</i>	I-1
<i>INTERDICTION_ZONE</i>	I-1
<i>COALESCED_FLAG</i>	I-1
<i>MOVEMENT_FLAG</i>	I-1
<i>TGT_HARDNESS</i>	I-1
<i>TGT_POP_DAY</i>	I-1
<i>INTEL_INDEX</i>	F-4
<i>UNIT_X_COORD</i>	F-4
<i>UNIT_Y_COORD</i>	F-4
<i>DEF_INDEX</i>	F-4
<i>DELAY_HOURS</i>	F-4
<i>CURRENT_ACTION</i>	C-1
<i>DESTINATION_X_COORD</i>	F-4
<i>DESTINATION_Y_COORD</i>	F-4
<i>DESTINATION_UNIT_ID</i>	I-2
<i>ENGAGEMENT_DEPTH</i>	F-4
<i>ENGAGEMENT_INDEX</i>	F-4
<i>AIR_CASUALTIES</i>	F-4
<i>TOTAL_CASUALTIES</i>	F-4
<i>MOVEMENT_DIRECTION</i>	C-1
<i>COALESCED_UNIT_ID</i>	I-2
<i>TGT_EFFECTIVENESS</i>	F-4

Table 16. Land Unit Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>COMMAND_UNIT_ID</i>	I-2
<i>SUBORDINATE_UNIT_ID</i>	I-2

Table 17. Land Unit Commands Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>UNIT_ID</i>	I-2
<i>TIED_UNIT_ID</i>	I-2

Table 18. Tied Land Units Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>UNIT_ID</i>	I-2
<i>EXERCISE_DAY</i>	I-1
<i>CYCLE</i>	I-1
<i>ENEMY_UNIT_ID</i>	I-2
<i>CMBT_PWR_DESTROYED</i>	F-4
<i>AAA_DESTROYED</i>	F-4
<i>TANKS_DESTROYED</i>	F-4
<i>APCS_DESTROYED</i>	F-4
<i>ARTY_DESTROYED</i>	F-4

Table 19. Land Unit Engagement Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>UNIT_ID</i>	I-2
<i>ACTION_TYPE</i>	C-1

Table 20. Valid Land Unit Actions Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>UNIT_ID</i>	I-2
<i>VEHICLE_TYPE</i>	C-4
<i>QUANTITY</i>	I-2

Table 21. Land Unit Vehicles Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>ORDER_NUMBER</i>	I-2
<i>EXERCISE_DAY</i>	I-1
<i>CYCLE</i>	I-1
<i>ORDER</i>	C-1
<i>UNIT_ID</i>	I-2
<i>QUADRANT_X_COORD</i>	F-4
<i>QUADRANT_Y_COORD</i>	F-4
<i>DESTINATION_UNIT_ID</i>	I-2
<i>DESTINATION_X_COORD</i>	F-4
<i>DESTINATION_Y_COORD</i>	F-4

Table 22. Land Unit and Quadrants Orders Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>SEMINAR_NUMBER</i>	I-1
<i>PASSWORD</i>	C-8
<i>CURRENT_DAY</i>	I-1
<i>MAX_NUMBER_CYCLES</i>	I-1
<i>MAX_NUMBER_AC</i>	I-2
<i>MAX_NUMBER_AIRBASES</i>	I-2
<i>MAX_NUMBER_MUNITIONS</i>	I-2
<i>MAX_NUMBER_PCL</i>	I-2
<i>MAX_NUM_CORPS</i>	I-1
<i>RERUN_DAYS</i>	I-1
<i>MAX_PREDIRECT</i>	I-4
<i>AIRLIFT_TONS_AVAIL</i>	F-4
<i>AIRLIFT_TONS_USED</i>	F-4
<i>SURFACE_TONS_AVAIL</i>	F-4
<i>SURFACE_TONS_USED</i>	F-4
<i>BASE_LOAD_FAC_FLAG</i>	I-1
<i>2ATAF_FLAG</i>	I-1
<i>4ATAF_FLAG</i>	I-1
<i>AAFCE_REROLE_FLAG</i>	I-1
<i>AAFCE_AC_MOVE_FLAG</i>	I-1
<i>AAFCE_LOG_AIR_FLAG</i>	I-1
<i>SORTIES_FLAG</i>	I-1
<i>LOCKOUT_FLAG</i>	I-1
<i>REPORT_FLAGS</i>	I-1

Table 23. Seminar Control Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>QUAD.X.COORD</i>	I-1
<i>QUAD.Y.COORD</i>	I-1
<i>WX_ZONE</i>	I-1
<i>RED.PREPARED.DEF</i>	I-1
<i>BLUE.PREPARED.DEF</i>	I-1
<i>TERRAIN.TYPE</i>	C-1
<i>OWNED.BY</i>	I-1

Table 24. Quadrants Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>WX_ZONE</i>	I-1
<i>EXECISE.DAY</i>	I-1
<i>CYCLE</i>	I-1
<i>PROB.GOOD.WX</i>	F-4
<i>PROB.BAD.WX</i>	F-4

Table 25. Weather Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>ATAF</i>	I-1
<i>MISSION.NUMBER</i>	I-2
<i>EXERCISE.DAY</i>	I-1
<i>CYCLE</i>	I-1
<i>AC.NAME</i>	C-5
<i>AC.ROLE</i>	C-1
<i>MISSION.TYPE</i>	I-1
<i>NUMBER.SORTIES</i>	I-2
<i>REGULAR.ABORTS</i>	I-2
<i>PMS.ABORTS</i>	I-2
<i>WX.ABORTS</i>	I-2
<i>JETTISONS</i>	I-2
<i>LOSSES.ENE.GA</i>	I-2
<i>LOSSES.ENE.CAP</i>	I-2
<i>LOSSES.ENE.ESC</i>	I-2
<i>LOSSES.ENE.DCA</i>	I-2
<i>LOSSES.ENE.OTHER</i>	I-2

Table 26. Air Mission Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>AC_NAME</i>	C-5
<i>AC_ROLE</i>	C-1
<i>MISSION_TYPE</i>	I-1

Table 27. Valid Mission Type - A/C Role Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>ATAF</i>	I-1
<i>MISSION_NUMBER</i>	I-2
<i>EXERCISE_DAY</i>	I-1
<i>CYCLE</i>	I-1
<i>AC_NAME</i>	C-5
<i>AC_ROLE</i>	C-1
<i>ESCORT_TYPE</i>	I-1
<i>NUMBER_SORTIES</i>	I-2
<i>REGULAR_ABORTS</i>	I-2
<i>PMS_ABORTS</i>	I-2
<i>WX_ABORTS</i>	I-2
<i>JETTISONS</i>	I-2
<i>LOSSES_ENE_GA</i>	I-2
<i>LOSSES_ENE_AA</i>	I-2

Table 28. OCA, IND, BAI, REC Escort Air Mission Relations

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>ATAF</i>	I-1
<i>MISSION_NUMBER</i>	I-2
<i>EXERCISE_DAY</i>	I-1
<i>CYCLE</i>	I-1
<i>ENEMY_AB_ID</i>	I-1
<i>AC_DESTROYED</i>	I-2
<i>PERCENT_POL_DES</i>	I-1
<i>PERCENT_MUNITIONS_DES</i>	I-1

Table 29. OCA Air Mission Targets Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>ATAF</i>	I-1
<i>MISSION_NUMBER</i>	I-2
<i>EXERCISE_DAY</i>	I-1
<i>CYCLE</i>	I-1
<i>CORPS_ID</i>	I-1
<i>CMBT_PWR_DESTROYED</i>	F-4
<i>DELAY_FACTOR</i>	F-4

Table 30. CAS Supports Corps Air Mission Relation

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>SIDE</i>	I-1
<i>ATAF</i>	I-1
<i>MISSION_NUMBER</i>	I-2
<i>EXERCISE_DAY</i>	I-1
<i>CYCLE</i>	I-1
<i>ENEMY_UNIT_ID</i>	I-1
<i>TANKS_DESTROYED</i>	I-2
<i>APCS_DESTROYED</i>	I-2
<i>ARTY_DESTROYED</i>	I-2
<i>AAA_DESTROYED</i>	I-2
<i>CMBT_PWR_DESTROYED</i>	F-4
<i>DELAY_FACTOR</i>	F-4

Table 31. BAI and IND Targets Air Mission Relations

<i>COLUMN NAME</i>	<i>FORMAT</i>
<i>CONST_NAME</i>	C-18
<i>ARRAY_INDEX</i>	I-2
<i>VALUE</i>	F-4

Table 32. Air and Land Constants Relations

Bibliography

1. *Ingres Reference Manual*. Relational Technology Inc., Alameda, California, 1986.
2. *Theater Warfare Exercise Maintenance Manual*. Air University Center for Aerospace Doctrine, Research and Education, CADRE/EDW, Maxwell AFB, Alabama, 1982.
3. *Theater Warfare Exercise Users' Handbook*. Air University Center for Aerospace Doctrine, Research and Education, CADRE/EDW, Maxwell AFB, Alabama, 1986.
4. Robert H. Bonczek et al. *Micro Database Management*. Academic Press, Orlando, FL, 1984.
5. I. T. Hawryszkiewicz. *Database Analysis and Design*. Science Research Associates, Chicago, 1984.
6. D. R. Howe. *Data Analysis for Data Base Design*. Thomson Litho Ltd, Kilbride, Scotland, 1983.
7. Henry F. Korth and Abraham Silberschatz. *Database System Concepts*. MacGraw-Hill, New York, 1986.
8. Mark Kross. *Developing a User Interface for the Theater War Exercise*. Master's thesis, Air Force Institute of Technology, 1987.
9. Philippe G. Lehot and Stephen H. Kaisler. *Network Database Systems*. Architecture Technology Corp, Minneapolis, 1984.
10. Software Digest Ratings Newsletter. *Relational Database Management Programs for IBM, Compaq, and Other Compatible Personal Computers*. March 1986.
11. Captain Mark A. Roth. AFIT/ENG Wright-Patterson AFB OH. Personal interviews. June-Aug 1987.
12. BIS Applied Systems. *Data Base Techniques, Software Selection and Systems Development*. Q.E.D. Information Sciences, Wellesley, MA, 1980.
13. Toby J. Teorey et al. A logical design methodology for relational databases using the extended entity-relationship model. *ACM Computing Surveys*, 18(2), June 1986.

Vita

Capt Michael D. Brooks was born on 8 February, 1954 in Charleston, South Carolina. He graduated from Middleton High School in 1973 and enlisted in the U.S. Air Force the same year. He was selected for the Airman Education and Commissioning Program in 1976 and graduated from the University of South Carolina in 1979 with a Bachelor of Science in Computer Science. Upon graduation, he attended Officer Training School and was commissioned in April, 1980. After commissioning, he was assigned to the 552 Airborne Warning and Control Division at Tinker AFB, OK where he served as a systems analyst in the area of E-3A (AWACS) command and control software. In 1984, he was transferred to the Tactical Air Warfare Center at Eglin AFB, FL where he served as Chief, Intelligence Software Support Branch, in the Information Systems Directorate until entering the Air Force Institute of Technology, School of Engineering, in June 1986.

Permanent address: 206 Longleaf Drive
Summerville, SC 29483

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT / GCS / ENG / 87D-6			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL (if applicable) AFIT/ENG		7a. NAME OF MONITORING ORGANIZATION		
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433			7b. ADDRESS (City, State, and ZIP Code)		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (if applicable)		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS (City, State, and ZIP Code)			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
			WORK UNIT ACCESSION NO.		
11. TITLE (Include Security Classification) See box 19					
12. PERSONAL AUTHOR(S) Michael D. Brooks, B.S., Capt, USAF					
13a. TYPE OF REPORT MS Thesis	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 1987 December		15. PAGE COUNT 82	
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Data Bases, Simulation, User Needs		
05	02				
14	02				
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
<p>Title: Developing a Database System and Air Simulation Software for the Theater War Exercise</p> <p>Thesis Chairman: Mark A. Roth, Captain, USAF</p>					
<p>Approved for public release: IAW AFR 190-4 Lynn E. Wolsten, Development Air Force Institute of Technology Wright-Patterson AFB, OH 45433</p>					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Michael D. Brooks, Captain, USAF			22b. TELEPHONE (Include Area Code) 513-255-3576		22c. OFFICE SYMBOL AFIT/ENG

19.

The Theater War Exercise (TWX) is a computer assisted, theater level, airpower employment exercise conducted by the Air Force Wargaming Center. TWX is designed to provide senior Air Force officers with a realistic, five day, European theater conflict which will test and evaluate their warfighting skills.

Presently, the exercise runs on a Honeywell H6000 series computer with all user inputs made on slow, hard copy devices which severely limits the interaction by controllers and exercise participants. Also, due to an application dependent file management system and hardcoded simulation constraints, the exercise is difficult to maintain and enhance.

The goal of this thesis effort was to bring the TWX system up to modern standards in both hardware and software capabilities while providing better exercise portability. This was accomplished by incorporating a relational database management system (Ingres) to provide increased data flexibility and a fourth generation language (4GL) to develop a new user interface. Also, to provide better portability, the exercise was re-hosted to a PC compatible microcomputer / DEC MicroVAX II configuration.

An integrated database design approach, incorporating both top-down and bottom-up relational design methods, was used to develop the new TWX database. With this approach, conceptual design tools such as entity-relationship diagrams have been combined with the traditional data dependency and normalization design approach. Finally, the Fortran air battle simulation was rewritten to incorporate the Ingres DBMS embedded database calls to access the new TWX database.

By incorporating a commercial database system, improving the interface, and re-hosting the exercise to a microcomputer environment, the new Micro-TWX system is a significant improvement over the current Honeywell system and will be much easier to maintain and enhance.